On $(1, \varepsilon)$-Restricted Assignment Makespan Minimization

Deeparnab Chakrabarty* Sanjeev Khanna† Shi Li‡

Abstract

Makespan minimization on unrelated machines is a classic problem in approximation algorithms. No polynomial time $(2-\delta)$-approximation algorithm is known for the problem for constant $\delta > 0$. This is true even for certain special cases, most notably the *restricted assignment* problem where each job has the same load on any machine but can be assigned to one from a specified subset. Recently in a breakthrough result, Svensson [16] proved that the integrality gap of a certain configuration LP relaxation is upper bounded by 1.95 for the restricted assignment problem; however, the rounding algorithm is *not known* to run in polynomial time.

In this paper we consider the $(1, \varepsilon)$-restricted assignment problem where each job is either heavy ($p_j = 1$) or light ($p_j = \varepsilon$), for some parameter $\varepsilon > 0$. Our main result is a $(2 - \delta)$-approximate *polynomial time* algorithm for the $(1, \varepsilon)$-restricted assignment problem for a fixed constant $\delta > 0$. Even for this special case, the best polynomial-time approximation factor known so far is 2. We obtain this result by rounding the configuration LP relaxation for this problem. A simple reduction from vertex cover shows that this special case remains NP-hard to approximate to within a factor better than 7/6.

1 Introduction

In the makespan minimization problem, we are given a set M of m machines, and a set J of n jobs where a job j contributes a load of p_{ij} to a machine i, if assigned to it. The goal is to assign each job to a machine, so that the maximum load on any machine is minimized. Formally, one seeks an allocation $\sigma : J \to M$ minimizing $\max_{i \in M} \sum_{j:\sigma(j)=i} p_{ij}$. In 1990, Lenstra, Shmoys, and Tardos [13] gave a 2-approximation for the problem, and showed that it is NP-hard to obtain an approximation factor better than 3/2. Closing this gap is an outstanding problem in approximation algorithms.

In order to understand the problem better, researchers have focused on special cases. The most notable among them is the *restricted assignment* problem. In this problem, each job $j \in J$ has an inherent load p_j but it can be assigned only to a machine from a specified subset. Equivalently, each $p_{ij} \in \{p_j, \infty\}$ with $p_{ij} = \infty$ for machines i which j cannot be assigned to. The hardness of 3/2 carries over to the restricted assignment problem and no polynomial time $(2 - \delta)$-algorithm is known for any constant $\delta > 0$. In a breakthrough, Svensson [16] proved that the integrality gap of a certain configuration LP for the restricted assignment problem is at most 33/17. Svensson's result can thus be used to efficiently *estimate the value* of the optimum makespan to within a factor of 33/17; however, no *polynomial time* algorithm to compute a corresponding schedule is known. Nonetheless, it gives hope[1] that the restricted assignment case may have a polynomial time 'better-than-factor-2' algorithm.

Our paper makes progress on this front. We study the $(1, \varepsilon)$-restricted assignment problem, in which all jobs fall in two classes: heavy or light. Every heavy job has load, $p_j = 1$, and each light job has $p_j = \varepsilon$, for some parameter $\varepsilon > 0$, and the goal is to find a schedule which minimizes the makespan. We give a $(2 - \delta^*)$-approximate *polynomial time* algorithm for the $(1, \varepsilon)$-restricted assignment problem for a constant $\delta^* > 0$.

The $(1, \varepsilon)$-case is interesting because in some sense it is the simplest case which we do not understand. If all jobs have the same size, the problem becomes a matching problem. If there are two job sizes, we can assume they are 1 and $\varepsilon < 1$ by scaling. The reader should think of ε as a number that tends to 0, as there is a simple $(2 - \varepsilon)$-approximation if each job has size either 1 or ε (see Appendix A). The $(1, \varepsilon)$-restricted assignment problem is already hard – it is NP hard to obtain an approximation factor $< 7/6$ for this problem (see Appendix B), and no $(2-\delta)$-approximation is known for any δ independent of ε. It is plausible that an understanding of the $(1, \varepsilon)$-case might lead to an understanding of the restricted assignment case; indeed,

*Microsoft Research. dechakr@microsoft.com

†Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104. Email: sanjeev@cis.upenn.edu. Supported in part by National Science Foundation grant CCF-1116961.

‡TTIC. shili@ttic.edu

[1] Without discussing this technically, we refer the reader to the articles by Feige [9], and Feige and Jozeph [10].

Svensson [16], in his paper, first gives an improved integrality gap of $(5/3+\varepsilon)$ for this special case before proving his result for the general case.

THEOREM 1.1. *(Main Result.) There is a polynomial time $(2-\delta^*)$-approximation algorithm, where $\delta^* > 0$ is a specific constant, for makespan minimization in the $(1,\varepsilon)$-restricted assignment case.*

1.1 Our Techniques For concreteness, let us assume for now that the optimal makespan is 1. Note that once we have an assignment of the heavy jobs to machines, we can decide in polynomial time whether there is a (fractional) allocation of light jobs so that the total makespan is at most $(2-\delta)$ or not, for any $\delta > 0$. Such an assignment of heavy jobs is called a δ-*good* assignment, and given such an assignment one can recover an integral assignment for light jobs as well such that the total load on any machine is at most $(2-\delta+\varepsilon)$. The rounding process to recover a δ-good assignment proceeds in three phases.

In the first phase, we 'reduce' our instance to a *canonical instance* where for each heavy job there is a distinct (private) set of machines to which it can be assigned to, and for each unassigned light job, there are at most two machines to which it can be assigned to. Such a pre-processing technique has also been used in tackling the max-min allocation problem [4, 6]. There are two main parameters of interest in a canonical instance, namely, a parameter p that asserts that the positive fractional assignment of a heavy job to a machine is at least $1/p$, and a parameter q, that asserts that the total load of light jobs shared by any two machines is either 0 or is at least $1/q$.

The second phase of the rounding process is a *coarsening* of the parameters p and q of the canonical instance where we ensure that whenever a heavy job is fractionally assigned to a machine, the assignment is sufficiently large (at least $1/q_0$ for some constant q_0). Furthermore, the total light load shared by any pair machines is either 0 or at least $1/q_0$. The flip side is the total fractional load on a single machine could increase from 1 to roughly $1+\sqrt{\log q_0/q_0}$. The technique used to achieve this coarsening is essentially due to Feige [8] in his work on max-min allocation, and is done by iteratively assigning heavy jobs and 'merging' light jobs. The proof uses the asymmetric Lovasz Local Lemma (LLL), and polynomial time algorithms to do the same are guaranteed by the recent works of Moser and Tardos [14] and Haeupler, Saha, and Srinivasan [11].

The heart of our approach and our main technical contribution is the third and the final phase of rounding. At this point we have a canonical instance where each heavy job is assigned to a constant number q_0 of machines that are private to it, each light job has constant size, and is shared between at most two machines. Note that if the fractional load on each machine was at most 1, then things are trivial – assign the heavy job to the machine which is fractionally assigned more than $1/q_0$ of it, and the total load on it become $2-1/q_0$. However, the second step has increased the fractional load from 1 to $1+\sqrt{\log q_0/q_0}$, and this 'extra' load swamps the gain of $1/q_0$. This issue does not arise in the max-min allocation where one targets a constant factor; however, it defeats our goal of beating the factor 2-approximation for makespan minimization.

Nevertheless, if we could find an assignment such that the total light load on any machine receiving a heavy job is at most $1-1/\text{polylog}q_0$, then we are in good shape, and this is what we do. We find such an assignment by randomized rounding and again use the (asymmetric) LLL. A key and difficult aspect of making this entire approach work is to have only a small degree of dependence between various "bad" events in the final rounding step. This reduction in dependence is the essence of our approach, and is accomplished by the structure of canonical instances, and further simplifying this structure before picking the final random assignment.

1.2 Relevant Related Work We briefly note some other works on makespan minimization. Ebenlendr, Krcál, and Sgall [7] consider the special case of the restricted assignment makespan minimization problem where each job could be assigned to at most 2 machines, and design a polynomial time 1.75-approximation algorithm for the same. Interestingly, even when jobs can go to at most two machines, the general makespan minimization problem seems difficult; Verschae and Wiese [17] show that the configurational LP has integrality gap tending to 2. Kolliopoulos and Moysoglou [12] consider the restricted assignment problem with two jobs sizes as well; they show that Svensson's estimation algorithm can be improved to 1.883 for this case. See Appendix A for a slightly better factor.

The 'dual' problem to makespan minimization, max-min allocation, where jobs need to be allocated to maximize the minimum load, has seen interesting developments recently. A constant factor approximation is known for the 'restricted assignment' case (the so-called *Santa Claus* problem), where each job has the same load for all the machines it can go to. This follows from the works of Bansal and Sviridenko [4], Feige [8], and the constructive LLL version due to [14, 11]. Our work closely follows this line and exhibits its utility for the makespan minimization problem. Another line of work on the Santa Claus problem is via local search; Asad-

pour, Feige, and Saberi [2] show an upper bound of 4 on the integrality gap via a not-known-to-be-polynomial time local search algorithm. Polacek and Svensson [15] use these ideas to give a *quasipolynomial* time, $4+\varepsilon$-approximation. Very recently, Annamalai, Kalaitzis, and Svensson [1] improve this to get a polynomial time 13-approximation. For the *general* max-min allocation problem, Chakrabarty, Chuzhoy, and Khanna [6], improving upon earlier results by Asadpour and Saberi [3] and Bateni, Charikar, and Guruswami [5], give a $O(n^{\varepsilon})$-approximation algorithm which runs in $O(n^{1/\varepsilon})$-time.

2 Linear Programming Relaxation

Recall that we denote the set of all machines by M and the set of all jobs by J. We assume that J is partitioned into the set of heavy jobs by J_{H}, and the set of light jobs by J_{L}. We consistently use j to index jobs, and i, h and k to index machines. For any job $j \in J$, we denote by M_j the set of machines to which job j can be assigned.

Given a guess $T \geq 1$ for the optimal makespan, the *configuration LP* w.r.t. T is as follows. For every machine i, $\mathscr{C}_i$ contains subsets of jobs of total load at most T which can be assigned to i . We have a variable $y_{i,C}$ for each machine i and subset $C \in \mathscr{C}_i$.

$$\text{(Conf LP 1)} \qquad \sum_{C\in\mathscr{C}_i} y_{i,C} = 1 \qquad \forall i \in M$$

$$\text{(Conf LP 2)} \qquad \sum_{i\in M}\sum_{C\in\mathscr{C}_i : j\in C} y_{i,C} = 1 \qquad \forall j \in J$$

Given an instance I, we let OPT_f be the *smallest* T for which the configuration LP has a feasible solution; OPT_f can be found by binary search. Trivially, $1 \leq \mathsf{OPT}_f \leq \mathsf{OPT}$, where OPT denotes the optimal (integral) makespan.

In this paper, we use the following simpler parametrized LP relaxation $\mathsf{LP}(\rho,\delta)$ tailored for $(1,\varepsilon)$-instances.

$$\sum_{i\in M_j} x_{i,j} = 1 \qquad \forall j \in J \tag{2.1}$$

$$\sum_{j\in J_{\mathsf{H}} : i\in M_j} x_{i,j} = z_i \qquad \forall i \in M \tag{2.2}$$

$$z_i \leq 1 \qquad \forall i \in M \tag{2.3}$$

$$z_i + \varepsilon \sum_{j\in J_{\mathsf{L}} : i\in M_j} x_{i,j} \leq 1+\rho\delta \qquad \forall i \in M \tag{2.4}$$

$$(1-\rho)z_i + x_{i,j} \leq 1 \qquad \forall j \in J_{\mathsf{L}}, i \in M_j \tag{2.5}$$

$$x_{i,j}, z_i \geq 0 \qquad \forall j \in J, i \in M_j$$

To get some intuition, consider the LP with $\rho = \delta = 0$. We claim there exists a feasible solution if $\mathsf{OPT} = \mathsf{OPT}_f = 1$. In this case, it must be that every machine either gets one heavy job or at most $\lfloor 1/\varepsilon \rfloor$ light jobs. In particular, any machine getting a light job *cannot* get a heavy job. Constraint (2.5) encodes this.

The connection between $\mathsf{LP}(\rho,\delta)$ and the configuration LP is captured by the following lemma.

LEMMA 2.1. *Given an $(1,\varepsilon)$-restricted assignment instance I with $\mathsf{OPT}_f \leq 1+\rho\delta$, there is a polynomial time procedure which returns another $(1,\varepsilon)$-instance I' which has a feasible solution to $\mathsf{LP}(\rho,\delta)$. Furthermore, given a schedule for I' with makespan T, the procedure returns a schedule for I of makespan $\leq T+\delta$.*

Proof. Let y be the solution to the configuration LP at $\mathsf{OPT}_f \leq 1+\rho\delta$. Call a configuration C *heavy* if it contains a heavy job. Define $z_i := \sum_{C \text{ is heavy}} y_{i,C}$ for all i, and $x_{i,j} := \sum_{C: j\in C} y_{i,C}$ for all i, j. Note that for all $i \in M$, we have $z_i + \varepsilon \sum_{j\in J_{\mathsf{L}}} x_{i,j} \leq 1+\rho\delta$ since each configuration has load $\leq 1+\rho\delta$.

Now, if for some light job j, $\sum_{C \text{ heavy}: j\in C} y_{i,C} > \rho z_i$, we remove j from J_{L} and set $\sigma(j) = i$. Let J'_{L} be the set of remaining jobs. The new instance is $\mathsf{I}' = (M, J_{\mathsf{H}} \cup J'_{\mathsf{L}})$. For any job j in J'_{L}, $x_{i,j} \leq \sum_{C:\text{not heavy}} y_{i,C} + \rho z_i = 1-(1-\rho)z_i$, that is, $(1-\rho)z_i + x_{i,j} \leq 1$. Thus, (z,x) is a feasible solution for $\mathsf{LP}(\rho,\delta)$ for I'. Now, given an assignment of jobs for I', we augment it to get one for I by assigning job $j \in J_{\mathsf{L}} \setminus J'_{\mathsf{L}}$ to $\sigma(j)$. Note

$$\begin{aligned}\varepsilon \left|\sigma^{-1}(i)\right| &< \varepsilon \sum_{j \text{ light}} \sum_{C \text{ heavy} : j\in C} \frac{y_{i,C}}{\rho z_i} \\ &= \frac{\varepsilon}{\rho} \sum_{C:\text{ heavy}} \frac{y_{i,C}}{z_i} |C \cap J_{\mathsf{L}}| \leq \delta\end{aligned}$$

since $\varepsilon |C \cap J_{\mathsf{L}}| \leq \rho\delta$ for heavy C.

The remainder of the paper is devoted to proving the following theorem.

THEOREM 2.1. *There is a constant $\delta_0 \in (0,1)$, such that given an instance I and a feasible solution to $\mathsf{LP}(\rho = 0.6, \delta_0)$, in polynomial time one can obtain a schedule for I of makespan at most $(2-2\delta_0)$.*

We conclude this section by showing that the preceding theorem suffices to establish our main result.

Proof of Theorem 1.1. Set $\delta^* = \delta_0/2$, where δ_0 is the constant specified in Theorem 2.1. Fix an instance I and the corresponding OPT_f. If $\mathsf{OPT}_f > 1+\rho\delta_0$, then the classic result of Lenstra et al. [13] returns a schedule whose makespan is at most $\mathsf{OPT}_f + 1 \leq \mathsf{OPT}_f \left(1 + \frac{1}{1+\rho\delta_0}\right) \leq (2-\delta^*)\,\mathsf{OPT}_f$. If $\mathsf{OPT}_f \leq 1+\rho\delta_0$, then Lemma 2.1 can be used to get an instance I' for which $\mathsf{LP}(\rho,\delta_0)$ is feasible. Theorem 2.1 returns a schedule for I' with makespan at most $(2-2\delta_0)$, which when fed to Lemma 2.1 gives a schedule for I of makesan at most $(2-\delta_0) \leq (2-\delta^*)\mathsf{OPT}_f$ since $\mathsf{OPT}_f \geq 1$. This proves Theorem 1.1.

3 Canonical Instances and δ-good Assignments

In this section we introduce the notion of canonical instances and formalize the notion of a δ-good assignment of heavy jobs for these instances.

In a canonical instance, heavy jobs have size $p_j = 1$. Light jobs can be scheduled fractionally and any light job can be assigned to at most two machines. Thus each light job is of type-(h,k) for some $h,k \in M$; it can only be assigned to h or k. It is possible that $h = k$; when $h \neq k$, (h,k) and (k,h) are two different job types. Subsequently, it will be clear that these types are differentiated and defined by how the LP assigns the jobs; for now the reader may think of $w_{h,k}$ as the load of light jobs which 'belong to k but can be assigned to h if k gets a heavy job'. Given h,k, we will merge the light jobs of type-(h,k) into a single job of total size equal to the sum of the light jobs. We call this the light load of type-(h,k). Henceforth, we use "light load" instead of "light jobs" in a canonical instance.

DEFINITION 3.1. *A* **canonical instance** *is defined by a triple* $(\{M_j : J \in J_{\mathsf{H}}\}, w, z)$*, where*

- *(A1) for every heavy job* $j \in J_{\mathsf{H}}$*,* $M_j \subseteq M$ *is the set of machines that* j *can be assigned to; for any pair heavy jobs* $j \neq j'$*, we have* $M_j \cap M_{j'} = \emptyset$*;*
- *(A2)* $w \in \mathbb{R}_{\geq 0}^{M \times M}$ *a matrix, where* $w_{h,k}$ *is the light load of type-*(h,k)*. If* $z_k = 0$ *and* $h \neq k$*, then* $w_{h,k} = 0$*; if* $z_h > 0$ *then* $w_{h,h} = 0$*;*
- *(A3)* $z : M \mapsto [0, 0.4]$ *is a function on* M *where* $z_i = 0$ *if and only if* $i \notin \bigcup_{j \in J_{\mathsf{H}}} M_j$*.*

There is an intrinsic fractional solution defined by the function z. If $i \in M_j$ for some $j \in J_{\mathsf{H}}$, then z_i is the fraction of the heavy job j assigned to machine i. A heavy job may not be fully assigned, but we will ensure that a decent portion of it is. If $h \neq k$, $(1 - z_k)$ fraction the $w_{h,k}$ light load of type-(h,k) is assigned to k, and the remaining z_k fraction is assigned to h. The $w_{i,i}$ light load of type-(i,i) is fully assigned to i. Given a matrix $w \in \mathbb{R}^{M \times M}$, we use the notation $w_{A,B}$ for subsets $A, B \subseteq M$ to denote the sum $\sum_{h \in A, k \in B} w_{h,k}$.

DEFINITION 3.2. *The directed graph* $G_w = (M, \{(h,k) : h \neq k, w_{h,k} > 0\})$ *formed by the support of* w*, with self-loops removed, is called the* **light load graph**.

DEFINITION 3.3. *Given a canonical instance* $\mathsf{I} = (\{M_j : j \in J_{\mathsf{H}}\}, w, z)$*, we say that* I *is a* (p, q, θ)**-canonical instance** *for some* $p \geq 1, q \geq 1$ *and* $\theta \in [0, 0.2)$*, if it satisfies the following properties (in addition to Properties (A1) to (A3)):*

- *(B1) for each* $i \in M$*, either* $z_i = 0$*, or* $z_i \geq 1/p$*;*
- *(B2) for every* $h, k \in M$*, either* $w_{h,k} = 0$ *or* $w_{h,k} \geq 1/q$*;*
- *(B3)* $\sum_{i \in M_j} z_i \geq 0.2 - \theta$*, for every* $j \in J_{\mathsf{H}}$*;*
- *(B4)* $z_h + \sum_{k \in M} w_{k,h}(1 - z_h) + \sum_{k \in M} w_{h,k} z_k \leq 1 + \theta$*, for every machine* $h \in M$*.*

Property (B1) says that none of the heavy job assignments is too small. Property (B2) says that any positive load of some type is large. Property (B3) says that a 'decent' fraction of each heavy job j is assigned. Property (B4) says that the total load assigned to a machine $h \in M$ in the intrinsic fractional solution is bounded. Our goal is to find a valid assignment $f : J_{\mathsf{H}} \mapsto M$ of heavy jobs to machines which leaves "enough room" for the light loads. We say f is *valid* if $f(j) \in M_j$ for every $j \in J_{\mathsf{H}}$. This motivates the following definition.

DEFINITION 3.4. (δ-GOOD ASSIGNMENT) *Given a* (p, q, θ)*-canonical instance and a number* $\delta \in (0, 1)$*, a valid assignment* $f : J_{\mathsf{H}} \to M$ *for a canonical instance is* δ*-good if all the light loads can be* fractionally *assigned so that each machine has total load at most* $2 - \delta$*.*

Define $f(J_{\mathsf{H}}) = \{f(j) : j \in J_{\mathsf{H}}\}$. The following theorem (basically Hall's condition) is a characterization of good assignments.

THEOREM 3.1. *For a canonical instance, an assignment* f *of heavy jobs is a* δ*-good assignment if and only if for every subset* $S \subseteq M$*,*

$$|S \cap f(J_{\mathsf{H}})| + w_{S,S} \leq (2 - \delta)|S|. \tag{3.6}$$

Proof. We define an instance of the single-source single-sink network flow problem as follows. Construct a directed bipartite graph $H = (A, M, E_H)$, where edges are directed from A to M. For every $h, k \in M$ such that $w_{h,k} > 0$, we have a vertex $a_{h,k} \in A$ that is connected to h and k. All edges in E_H have infinite capacity. Now an assignment f is δ-good if and only if we can send flow from A to M in H such that: (1) Each vertex $a_{h,k} \in A$ sends exactly $w_{h,k}$ flow, and (2) Each machine $i \in M$ receives at most $1_{i \notin f(J_{\mathsf{H}})} + 1 - \delta$ flow, where $1_{i \notin f(J_{\mathsf{H}})}$ is 1 if $i \notin f(J_{\mathsf{H}})$ and 0 otherwise. By Hall's theorem, there is a feasible flow if and only if the following holds: $\sum_{a_{h,k} \in A'} w_{h,k} \leq \sum_{h \in M(A')} \left[1_{h \notin f(J_{\mathsf{H}})} + 1 - \delta\right], \forall A' \subseteq A$, where $M(A')$ is the set of vertices in M adjacent to A'. It is easy to see that for every $S \subseteq M$, it suffices to consider the maximal A' with $M(A') = S$. For this A', we have $\sum_{a_{h,k} \in A'} w_{h,k} = w_{S,S}$. Thus, the condition can be rewritten as $w_{S,S} \leq (2 - \delta)|S| - |S \cap f(J_{\mathsf{H}})|, \forall S \subseteq M$. This finishes the proof.

3.1 Roadmap of the Proof

We are now armed to precisely state the ingredients which make up the proof of Theorem 2.1. In §4, we show how to reduce any instance to a canonical instance. The precise theorem that we will prove is the following, where $m = |M|$ is the number of machines.

THEOREM 3.2. *Let $\delta > 0, \delta' \in (0,1)$ and $\rho = 0.6$. Given an instance I of the $(1,\varepsilon)$-restricted assignment problem with a feasible solution to $\mathsf{LP}(\rho,\delta)$, there is a polynomial time procedure to obtain an $(\frac{m}{\rho\delta}, 1/\varepsilon, \rho\delta)$-canonical instance I' such that any δ'-good assignment for I' implies a schedule of makespan at most $(2-\delta'+2\varepsilon)$ for I.*

In §5, we reduce a canonical instance to one with "small" p and q. More precisely, we prove the following.

THEOREM 3.3. *For some large enough constant q_0 the following is true. Given a (p,q,θ)-canonical instance I, in polynomial time we can obtain a $(q_0, q_0, \theta + 16\sqrt{\log q_0/q_0})$-canonical instance I' such that any δ-good assignment for I' is a $(\delta - 16\sqrt{\log q_0/q_0})$-good assignment for I, for every $\delta \in (16\sqrt{\log q_0/q_0}, 1)$.*

Finally, in §6, we show how given a (q_0, q_0, θ)-canonical instance we can 'round' it to a δ-good instance where δ is inverse *polylogarithmic* in the q parameter. Observe, from definition of canonical instances, $(1/q - \theta)$-good assignments are trivial.

THEOREM 3.4. *For some large enough constant C, every $q_0 \geq 100$ and every $\theta \in [0, \log^{-5} q_0/4C)$, the following is true. Given a (q_0, q_0, θ)-canonical instance I, there is a polynomial time procedure to obtain a $(\log^{-5} q_0/C - 4\theta)$-good assignment for I.*

Assuming the above theorems, the proof of Theorem 2.1 follows easily.

Proof of Theorem 2.1. Let C be as in Theorem 3.4, and choose q_0 such that $400\sqrt{\log q_0/q_0} \leq \log^{-5} q_0/C$. Let $\delta_0 := \log^{-5} q_0/6C$. Given a feasible solution to $\mathsf{LP}(\rho,\delta_0)$, we convert it to a $(\frac{m}{\rho\delta_0}, 1/\varepsilon, \rho\delta_0)$-canonical instance I using Theorem 3.2. Then using Theorem 3.3, we obtain a $(q_0, q_0, \rho\delta_0 + 16\sqrt{\log q_0/q_0})$-canonical instance I'. Given I', via Theorem 3.4, we obtain a δ-good assignment with $\delta = 6\delta_0 - 4\rho\delta_0 - 64\sqrt{\log q_0/q_0}$. This in turn implies a $(\delta - 16\sqrt{\log q_0/q_0} = 3.6\delta_0 - 80\sqrt{\log q_0/q_0})$-good assignment for I. By choice of parameters, this is a $(2\delta_0 + 2\varepsilon)$-good assignment when $\varepsilon \leq 0.2\delta_0$. By Theorem 3.2, this implies a schedule of makespan $(2 - 2\delta_0)$ which proves Theorem 2.1.

The rest of the paper proves the above theorems in §4, §5, and §6 respectively which can be read in any order.

4 Reduction to Canonical Instances

This section is devoted to proving Theorem 3.2.

THEOREM 4.1. *Let $\delta > 0, \delta' \in (0,1)$ and $\rho = 0.6$. Given an instance I of the $(1,\varepsilon)$-restricted assignment problem with a feasible solution to $\mathsf{LP}(\rho,\delta)$, there is a polynomial time procedure to obtain an $(\frac{m}{\rho\delta}, 1/\varepsilon, \rho\delta)$-canonical instance I' such that any δ'-good assignment for I' implies a schedule of makespan at most $(2-\delta'+2\varepsilon)$ for I.*

Let x be any feasible solution for $\mathsf{LP}(\rho = 0.6, \delta)$. The solution x defines the following weighted bipartite graph $H = (M, J, E_H, x)$: if $x_{i,j} > 0$ for some $i \in M, j \in J$, there is an edge $(i,j) \in E_H$ of weight $x_{i,j}$. We will create the desired canonical instance I' by performing the following sequence of transformation steps.

4.1 Processing Heavy Jobs

Without loss of generality, we can assume $H[M \cup J_{\mathsf{H}}]$ is a forest. Indeed, if there is an even cycle in the sub-graph, we can *rotate* the cycle as follows. Color the edges of the cycle alternately as red and black. Uniformly decrease the x values of red edges and increase x values of the black edges. Observe that Constraints (2.1) and (2.2) remain satisfied, and Constraints (2.3), (2.4) and (2.5) are untouched since z_i's and $x_{i,j}$'s for light jobs j did not change. Apply the operation until the x-value of some edge in the cycle becomes 0. By applying this operation repeatedly, we can guarantee that the graph $H[M \cup J_{\mathsf{H}}]$ is a forest. Some heavy jobs j may be completely assigned to a machine i; in this case, the edge (i,j) forms a two-node tree, since $z_i \leq 1$. We call such trees trivial.

We now further modify the instance so that each connected component in $H[M \cup J_{\mathsf{H}}]$ is a star, with center being a heavy job, and leafs being machines. Consider any nontrivial tree τ in the forest $H[M \cup J_{\mathsf{H}}]$. We root τ at an arbitrary heavy job. If the weight $x_{i,j}$ between any heavy job j in τ and its parent machine i is at most $1/2$, we remove the edge (i,j) from τ. After this operation, τ is possibly broken into many trees.

Now focus on one particular such tree τ'. Note the following facts about τ': (i) τ' is rooted at a heavy job j^*; (ii) every machine i in τ' has either 0 or 1 child since $x_{i,j} > 1/2$ for any child j of i in τ' and (2.3) holds; (iii) all leaves are machines since a heavy job can only be partially assigned to its parent. Thus, in τ', the number of heavy jobs is exactly the number of non-leaf-machines plus 1.

LEMMA 4.1. *Let L be the set of leaf-machines in τ'. Then $\sum_{i \in L} z_i \geq 1/2$.*

Proof. Suppose there are t heavy jobs in the tree τ'. Since we may remove an edge of weight at most $1/2$

connecting the root of τ' to its parent in τ, we have $\sum_{i\in M(\tau')} z_i \geq t - 1/2$, where $M(\tau')$ is the set of machines in τ'. Since $z_i \leq 1$ for each $i \in M(\tau') \setminus L$ and $|M(\tau')\setminus L| = t-1$, we have $\sum_{i\in L} z_i \geq t-1/2-(t-1) = 1/2$.

We assign heavy jobs in τ' to machines in τ' as follows. Each non-leaf machine of τ' is guaranteed to be assigned a heavy job. There is one extra heavy job left, and we assign it to a leaf machine. The following lemma shows that any leaf-machine can yield to a valid assignment for the heavy jobs.

LEMMA 4.2. *Let i be any leaf-machine in τ'. There is a valid assignment of heavy jobs in τ' to machines in τ' such that*

1. *Any non-leaf-machine is assigned exactly one heavy job;*
2. *i is assigned exactly one heavy job;*
3. *No heavy jobs are assigned to other leaf-machines.*

Proof. Focus on the path from the root of τ' to the leaf-machine i. We assign each heavy job in this path to its child in the path. For all the other heavy jobs, we assign them to their parents. It is easy to see this assignment satisfies all the conditions.

We now create a new set of heavy jobs to *replace* the heavy jobs in τ'. For each non-leaf machine i, we create a new heavy job j with $M_j = \{i\}$. We also create a new heavy job j with M_j being the set of leaf machines. By Lemma 4.2, a valid assignment for the new machines implies a valid assignment for the original machines. Notice that new created heavy jobs j have disjoint M_j. This is true even if we consider the new jobs for all trees τ' as the machines in these trees are disjoint. For every new created heavy job j, we have $\sum_{i\in M_j} z_i \geq 1/2$: if $M_j = \{i\}$ for a non-leaf machine i, then $z_i > 1/2$ as the weight of edge from i to its child has weight at least $1/2$; if M_j is the set of all leaves, then by Lemma 4.1, $\sum_{i\in M_j} z_i \geq 1/2$.

We have created a new set J'_{H} of heavy jobs for I and the sets $\{M_j : j \in J'_{\mathsf{H}}\}$ are disjoint. An assignment of these big jobs imply an assignment of J_{H} via Lemma 4.2. From now on we let $J_{\mathsf{H}} = J'_{\mathsf{H}}$ and only consider the set of new heavy jobs. Thus, Property (A1) is satisfied.

Since we haven't modified $x_{i,j}$ and z_i for $i \in M$ and $j \in J_{\mathsf{L}}$, Constraint (2.3), (2.4) and (2.5) are satisfied. Constraint (2.1) is satisfied for light jobs $j \in J_{\mathsf{L}}$. We did not define $x_{i,j}$'s for the new created heavy jobs $j \in J_{\mathsf{H}}$ and thus Constraint (2.1) for heavy jobs $j \in J_{\mathsf{H}}$ and Constraint (2.2) are meaningless and henceforth will be ignored.

We now scale down z_i by a factor of $1-\rho = 0.4$ for all machines in $i \in M$, then Constraint (2.5) is strengthened to

$$z_i + x_{i,j} \leq 1, \qquad \forall j \in J_{\mathsf{L}}, i \in M. \tag{4.7}$$

For every $j \in J_{\mathsf{H}}$, we have $\sum_{i\in M_j} z_i \geq 0.5(1-\rho) = 0.2$. Every z_i is between 0 and 0.4. Moreover, if for some $j \in J_{\mathsf{H}}$ and some $i \in M_j$ we have $z_i = 0$, we remove i from M_j. Then, Property (A3) holds and Property (B3) holds with $\theta = 0$.

4.2 Processing Light Jobs Now let H be the weighted bipartite graph between M and J_{L}. In this step, we make sure that each light job is fractionally assigned to exactly 2 machines. To achieve this, perform the rotation operations to cycles in H, as we did before for heavy jobs. Note that the rotation preserves the sum $\varepsilon \sum_{j\in\cap J_{\mathsf{L}}: i\in M_j} x_{i,j}$. In order to maintain Condition (4.7), we may not be able to break all cycles in H. We say an edge (i,j) in H is tight if $x_{i,j}+z_i = 1$. We can perform rotation operations so that the non-tight edges form a forest. Also, since $z_i \leq 0.4$ for all machines i, $x_{i,j} \geq 1 - 0.4 = 0.6$ for a tight edge (i,j). Thus, each light job j is incident to at most 1 tight edge.

For each non-singleton tree τ in the forest formed by the non-tight edges, we root τ at an arbitrary light job and *permanently* assign each *non-leaf* light job in τ arbitrarily to one of its children. These light jobs are then removed from J_{L}. Notice that each machine can get permanently assigned at most 1 light job during this process. Each unassigned light job in the tree τ is incident to exactly one non-tight edge (since it was a leaf).

Therefore, each remaining light job j must be one of the following. First, j can be completely assigned to some machine i (thus $x_{i,j} = 1$ and $z_i = 0$), then, we say j is of type-(i,i). Second, j maybe incident to two edges, one tight, the other non-tight. Let (k,j) be a tight edge and (h,j) be the other edge; then we say j is of type-(h,k). This lets us define the matrix w: we let $w_{h,k}$ be the total load of light jobs of type-(h,k), or equivalently, ε times the number of light jobs of type-(h,k). For every light job j of type-(h,k), $h \neq k$, we have $x_{h,j} = z_k$ and $x_{k,j} = 1 - z_k$. Thus w satisfies Property (A2) and Property (B2) with $q = 1/\varepsilon$. Property (B4) holds with $\theta = \rho\delta$ as the $z_h + \sum_{k\in M} w_{k,h}(1-z_h) + \sum_{k\in M} w_{h,k} z_k$ is exactly the total fractional load assigned to h which is at most $1+\rho\delta$ by (2.4)

Property (B1) holds for a sufficiently large number $p = \exp(\mathrm{poly}(n))$ as each z_i can be represented using polynomial number of bits. However, we would like to start with $p = m/(\delta\rho)$, where m is the number of machines. If $0 < z_i < \rho\delta/m$ for some $i \in M_j$,

we change z_i to 0 and remove i from M_j. Then, Property (B3) still holds for $\theta = \rho\delta/m \times m = \rho\delta$ as there are at most m machines. Thus, our canonical instance is $(m/\rho\delta, 1/\varepsilon, \rho\delta)$-canonical. This ends the proof of Theorem 3.2.

5 Reducing Parameters p and q in Canonical Instances

This section is devoted to proving Theorem 3.3. The proof is analogous to a similar theorem proved by Feige [8] for max-min allocations, and therefore we only provide a sketch in the main body. All omitted proofs from this section can be found in Appendix D.

THEOREM 5.1. *For some large enough constant q_0 the following is true. Given a (p, q, θ)-canonical instance I, in polynomial time we can obtain a $(q_0, q_0, \theta + 16\sqrt{\log q_0/q_0})$-canonical instance I' such that any δ-good assignment for I' is a $(\delta - 16\sqrt{\log q_0/q_0})$-good assignment for I, for every $\delta \in (16\sqrt{\log q_0/q_0}, 1)$.*

Using the characterization of δ-good assignment given in Theorem 3.1, we define a *δ-witness* as a pair of sets which rules out any δ-good assignment.

DEFINITION 5.1. (δ-WITNESS) *A pair (S, T) of subsets of machines is called a* **δ-witness** *if $T \subseteq S$ and*

$$|T| + w_{S,S} > (2 - \delta)|S|. \tag{5.8}$$

Moreover, we call a δ-witness (S, T) **connected** *if S is (weakly) connected in the light load graph G_w.*

CLAIM 5.1. *If (S, T) is a δ-witness, then there is a connected δ-witness $(\tilde{S}, \tilde{T})$, with $\tilde{S} \subseteq S$ and $\tilde{T} \subseteq T$.*

CLAIM 5.2. *f is a δ-good assignment iff for every connected δ-witness (S, T) of w, $T \not\subset f(J_{\mathsf{H}})$.*

Now, we prove two main lemmas for our algorithm that alternatively reduce q and p. Let I be a (p, q, θ)-canonical instance. If $q \geq \max\{p, q_0\}$, Lemma 5.1 reduces it to a $(p, q/2, \theta')$-canonical instance; if $p \geq \max\{q, q_0\}$, Lemma 5.2 reduces it to a $(p/2, q, \theta')$-canonical instance.

LEMMA 5.1. *Let $\mathsf{I} = (\{M_j : j \in J_{\mathsf{H}}\}, w, z)$ be a (p, q, θ)-canonical instance. Assume $q \geq \max\{p, q_0\}$. Then, we can find in polynomial time a (p, q', θ')-canonical instance $\mathsf{I}' = (\{M_j : j \in J_{\mathsf{H}}\}, w', z)$, such that any δ'-good assignment f for I' is δ-good for I, where $q' = q/2, \theta' = \theta + 8\sqrt{\log q/q}, \delta' = \delta + 8\sqrt{\log q/q}$.*

The proof follows from an application of asymmetric LLL. We want that Property (B2) holds for q'. We apply the following natural procedure. For each (h, k) such that $0 < w_{h,k} < 1/q' = 2/q$, we change $w_{h,k}$ to $1/q'$ with probability $q' w_{h,k}$ and to 0 with probability $1 - q' w_{h,k}$. We need to show that Property (B4) holds. Also, we need to show that any δ-witness in the original instance is also a δ'-witness in the new instance. We apply the asymmetric LLL (Theorem C.2) to show that all these properties can hold. The idea is that a bad event depending on many other bad events must have a smaller probability. The detailed proof is in Appendix D.1.

LEMMA 5.2. *Let $\mathsf{I} = (\{M_j : j \in J_{\mathsf{H}}\}, w, z)$ be a (p, q, θ)-canonical instance, where $p \geq \max\{q, q_0\}$. We can find in polynomial time a (p', q, θ')-canonical instance $\mathsf{I}' = (\{M_j : j \in J_{\mathsf{H}}\}, w, z')$ such that any δ-good solution f for I' is also δ-good for I, where $p' = p/2, \theta' = \theta + 8\sqrt{\log p/p}$.*

This lemma is by symmetric LLL. To guarantee that each positive z_i has $z_i \geq 1/p' = 2/p$, we apply the following natural process: if $1/p \leq z_i < 1/p'$, we change z_i to $1/p'$ with probability $p' z_i$, and to 0 with the probability $1 - p' z_i$. All bad events in the proof are local; they only depend on a few variables. Thus, a symmetric LLL suffices to prove the lemma. The detailed proof is in Appendix D.2.

To complete the proof of Theorem 3.3, we apply Lemma 5.1 and Lemma 5.2 repeatedly till we obtain a (q_0, q_0, θ)-canonical instance $\mathsf{I} = (\{M_j : j \in J_{\mathsf{H}}\}, w, z)$, where $\theta \leq \rho\delta_1 + 16\sqrt{\log q_0/q_0}$ with the guarantee that a δ-good solution to I implies a $(\delta - 16\sqrt{\log q_0/q_0})$-good solution to the original instance.

6 Solving Canonical Instances with Small Values of p and q

This section is devoted to proving Theorem 3.4.

THEOREM 6.1. *For some large enough constant C, every $q_0 \geq 100$ and every $\theta \in [0, \log^{-5} q_0/4C)$, the following is true. Given a (q_0, q_0, θ)-canonical instance I, there is a polynomial time procedure to obtain a $(\log^{-5} q_0/C - 4\theta)$-good assignment for I.*

For convenience, we will make $\theta = 0$ by scaling down the light load matrix w by a factor of $\frac{0.6}{0.6+\theta}$. After this operation, Property (B4) will hold with $\theta = 0$ as we have $z_h \leq 0.4$. Let $q = (0.6 + \theta)q_0/0.6$; then the new instance is $(q, q, 0)$-canonical except that Property (B3) only holds with right side being 0.19 instead of 0.2 (as $\theta \leq .01$ for large enough C). At the end, we can scale up light loads by $(0.6 + \theta)/0.6$. As each machine is assigned strictly less than 2 units of total load, the scaling will increase the light load on each machine by at most $\theta/0.6 \times 2 \leq 4\theta$.

Thus, we can assume $\theta = 0$ and focus on a $(q, q, 0)$-canonical instance $\mathsf{I} = (\{M_j : j \in J_{\mathsf{H}}\}, w, z)$ from now on. With $\theta = 0$, Property (B4) implies that $w_{M,h} \leq 1$ for every $h \in M$.

Given an assignment $f : J_{\mathsf{H}} \to M$, for convenience we use $X = f(J_{\mathsf{H}})$ to denote the set of machines that are assigned heavy jobs. We define the concept of a 'boundary' of a set which will be crucial in what follows.

DEFINITION 6.1. (BOUNDARY OF A SET) *Given a subset S of machines, we define its boundary as*

$$\mathtt{bnd}(S) = \sum_{h \in S} \sum_{k \notin S} \left(w_{k,h}(1 - z_h) + w_{h,k} z_k\right).$$

DEFINITION 6.2. *Let the* deficiency *of a machine $h \in M$ be $\phi_h = 1 - z_h - \sum_{k \in M} \left(w_{k,h}(1 - z_h) + w_{h,k} z_k\right)$. The deficiency of a subset $S \subseteq M$ is $\phi(S) = \sum_{h \in S} \phi_h$.*

Thus, $\phi_h \geq 0$ measures how far away Property (B4) is from being tight. With the definition, we can rewrite the condition for δ-good assignments. From Theorem 3.1, f is δ-good iff for every $S \subseteq M$, we have

$$w_{S,S} + |S \cap X| \leq (2 - \delta)|S|. \tag{6.9}$$

Adding the definition of ϕ_h for every $h \in S$, we get that $\phi(S) + z(S) + w_{S,S} + \mathtt{bnd}(S) = |S|$.

The left hand side of (6.9) is

$$\begin{aligned} &|S| + |S \cap X| - \phi(S) - z(S) - \mathtt{bnd}(S) \\ &= 2|S| - |S \setminus X| - (\phi(S) + z(S) + \mathtt{bnd}(S)). \end{aligned}$$

Thus, f is δ-good iff for every $S \subseteq M$ we have

$$|S \setminus X| + \phi(S) + z(S) + \mathtt{bnd}(S) \geq \delta|S|. \tag{6.10}$$

We fix $\delta_0 \in (0, 0.001)$ whose value will be decided later. We say a machine $h \in M$ is **green** if $\phi_h + z_h \geq \delta_0$ and **red** otherwise. Let R be the set of red machines.

To check if f is δ-good, we can check the Inequality (6.10) for every $S \subseteq M$. With some condition on X and some loss on the goodness, we only need to check the above condition for $S \subseteq X \cap R$. To be more specific,

LEMMA 6.1. *Let $c_1 \geq 1, \delta \in (0, 1)$. Suppose $f : J_{\mathsf{H}} \to M$ is a valid assignment with $X = f(J_{\mathsf{H}})$ satisfying*

- *(C1) for every $h \in M$, we have $\sum_{k \in X \cap R} w_{h,k} \leq c_1 \log q$;*
- *(C2) for every subset $T \subseteq X \cap R$, $\phi(T) + z(T) + \mathtt{bnd}(T) \geq \delta|T|$.*

Then f is a $\frac{\delta\delta_0}{2c_1 \log q}$-good assignment.

Proof. Decompose $S = P \cup Q$ where $P = S \cap X \cap R, Q = S \setminus (X \cap R)$. Each machine in Q contributes at least δ_0 to the left-side of Inequality (6.10). If $|Q| \geq \delta|S|/(2c_1 \log q)$, then the inequality with δ replaced with $\frac{\delta\delta_0}{2c_1 \log q}$ holds trivially.

So, assume $|Q| < \delta|S|/(2c_1 \log q)$. For large enough q, we get $|P| > 0.99|S|$. By Condition (C2) for T being P, we have $\phi(P) + z(P) + \mathtt{bnd}(P) \geq \delta|P|$. Notice that $\mathtt{bnd}(S) \geq \mathtt{bnd}(P) - \sum_{h \in P, k \in Q} \left(w_{k,h}(1 - z_h) + w_{h,k} z_k\right)$. Notice that for any machine $k \in Q$, $w(k, P) \leq c_1 \log q$ by Condition (C1) and $w(P, k) \leq w(M, k) \leq 1$. We have $\mathtt{bnd}(S) \geq \mathtt{bnd}(P) - |Q|(c_1 \log q + 1) \geq \mathtt{bnd}(P) - 0.51\delta|S|$. Thus, $\phi(S) + z(S) + \mathtt{bnd}(S) \geq \phi(P) + z(P) + \mathtt{bnd}(P) - 0.51\delta|S| \geq \delta|P| - 0.51\delta|S| \geq 0.48\delta|S|$.

To prove Theorem 3.4, it suffices to find an assignment which satisfies Properties (C1) and (C2) for suitable δ and c_1. In the remainder of this section, we will focus on finding such an assignment.

Sketch of the Proof. Suppose for the time being all the positive $w_{h,k}$'s are very close to 1. $w_{M,h} \leq 1$ implies that in the light load graph G_w any machine h can have in-degree at most 1. In other words, G_w is a collection of disjoint cycle-rooted trees. Suppose again there are no cycles, and so G_w is a collection of trees.

Now consider the random process which allocates job $j \in J_{\mathsf{H}}$ to machine i with probability proportional to z_i. We now describe some bad events such that if none of them occur we satisfy (C1) and (C2). The first bad event is of course the negation of (C1) (corresponding to $\mathscr{A}$ in what follows). The second bad event (corresponding to $\mathscr{B}$ in what follows) occurs if the forest induced by $X \cap R$ contains a connected component larger than $\Theta(\log q)$.

Note that if these bad events don't occur then any subset $S \subseteq X \cap R$ which does not contain roots of trees in the forest satisfy (C2) (for $\delta = \Theta(\log^{-1} q)$). This is because every connected component contributes 1 in-degree to $\mathtt{bnd}(S)$.

Now suppose S contains the root r as well. Since r has no incoming edges and is red, by (B4), it has many out-edges. The third bad event (corresponding to $\mathscr{C}$ in what follows) occurs if lots of out-neighbors of a machine have been picked in $X \cap R$. If $\mathscr{C}$'s doesn't occur in addition to the $\mathscr{B}$'s, then the out-edges of r which exit S contributes the boundary term in (C2). In summary, if no bad event of type $\mathscr{A}, \mathscr{B}, \mathscr{C}$ occur, then (C1) and (C2) are satisfied. The formal statement of this is Lemma 6.3.

To show that with positive probability none of the bad events occur, we invoke the asymmetric Lovasz Local Lemma. To do so, we need to argue about the dependency structure of the various events. We use a

connection to Galton-Watson branching processes described by Moser and Tardos to prove that $\mathscr{B}$ has 'good' dependency properties (the concrete statement is Lemma 6.4.). The algorithm follows from the constructive versions of LLL due to [14, 11].

Till now we have been assuming positive $w_{h,k}$'s are close to 1. In reality, we divide the edges into two classes: *dense*, if $w_{h,k} \geq (c_2 \log q)^{-1}$ and *sparse* otherwise. Note that dense edges no longer form trees. However, for each machine which has at least one dense edge coming into it (which are called *in-dense* later); we arbitrarily pick one of them and color it *red* and only count the red edges towards the boundary.

The reason such a 'sparsification' helps is that it decreases the dependence among events leading to the application of LLL. To take care of machines with only sparse in-coming edges, we need another type of bad events (corresponding to $\mathscr{D}$ in what follows) whose non-existence implies a large contribution to the boundary for such machines. This ends the sketch of the proof. We begin the details with some definitions.

DEFINITION 6.3. *An edge $(h,k) \in G_w$ is* **dense** *if $w_{h,k} \geq (c_2 \log q)^{-1}$, and* **sparse** *otherwise, where c_2 is large enough constant ($c_2 = 300$ suffices).*

DEFINITION 6.4. *A machine h is* **in-sparse** *if* all *incoming edges $(k,h) \in G_w$ are sparse. Otherwise, h is* **in-dense**.

DEFINITION 6.5. *A machine h is called* **out-dense** *if $\sum_{k:(h,k) \text{ is dense}} z_k \geq \frac{1}{c_3}$ where c_3 is a large enough constant ($c_3 = 200$ suffices). Otherwise, the machine is called* **out-sparse**.

We are now ready to describe our algorithm for assigning heavy jobs to machines satisfying (C1) and (C2). Our algorithm starts with a pre-processing of the fractional solution, and then recovers a good integral assignment of heavy jobs from the pre-processed solution using randomized rounding. note by $X = f(J_{\mathsf{H}})$ is the set of machines getting heavy jobs.

6.1 Pre-processing of the Instance For every in-dense, red machine $h \in M$, we arbitrarily select an incoming dense edge (k,h) of h. If k is red and out-sparse, we color the edge (k,h) in red. Every machine has at most one incoming red edge. Moreover, the two end points of a red edge are also red. Then each connected component formed by red edges is either a tree, or a tree plus an edge. Recall X is the set of machines which we will assign heavy jobs and R is the set of red machines.

We want to ensure that $G_w[X \cap R]$ does not contain any red cycles. That is, for any cycle of red edges in G_w, we wish to ensure that at least one of these machines is not assigned a heavy job. For each heavy job j, we now identify a subset $M'_j \subseteq M_j$ such that (i) $z(M'_j) \geq 0.49 z(M_j)$, and (ii) the subgraph of G_w induced by $\cup_{j \in J_{\mathsf{H}}} M'_j$ does not contain a red cycle.

We reduce the task of identifying M'_j to an instance of the generalized assignment problem where red cycles correspond to jobs and groups correspond to machines. If a red cycle C contains two machines from a group M_j, we can ignore it since one machine in the cycle will not get a heavy job. So assume C contains at most one job from each group M_j. The cost of assigning a red cycle C to a group M_j is z_h if some machine $h \in M_j$ participates in the cycle C and ∞ otherwise. Since each red cycle C contains at least two machines(if a red cycle contains one machine h, then $w_{h,h} > 0$, implying $z_h = 0$ by Property (A2) and thus h is not in any M_j), a solution that assigns each C uniformly to groups of all machines contained in C, is a feasible fractional solution where the load assigned to M_j is at most $z(M_j)/2$. We can now use the Lenstra-Shmyos-Tardos [13] algorithm to recover an integral assignment of red cycles to groups such that maximum load on any machine is at most the fractional load $z(M_j)/2$ plus the largest job size, which is at most $\max_{h \text{ red}} z_h \leq \delta_0$. Thus for any group M_j, the total z-value of machines chosen in the red cycle elimination step is at most $z(M_j)/2 + \delta_0 \leq 0.51 z(M_j)/$ since $\delta_0 \leq 0.001 \leq 0.01 z(M_j)$.

6.2 Randomized Assignment of Heavy Jobs and Bad Events Now we are ready to describe the randomized algorithm to get the heavy job assignment. For every heavy job j, assign it to a machine $f(j) := i \in M'_j$ with probability proportional to z_i. Note that the probability p_i a machine i gets a heavy job is at most $p_i \leq z_i/(0.49 \cdot 0.19) \leq 11 z_i$. We describe the bad events.

- Bad events $\mathscr{A}_h$, $h \in M$. $\mathscr{A}_h$ occurs if $\sum_{k \in X \cap R} w_{h,k} > c_1 \log q$. Setting $c_1 = 12$ is sufficient.
- Bad events $\mathscr{B}_T$, for a set T of $L = 10 \log q$ machines connected by red edges. $\mathscr{B}_T$ occurs if $T \subseteq X \cap R$.
- Bad events $\mathscr{C}_h$, $h \in M$. $\mathscr{C}_h$ occurs if $|\{k \in X \cap R : (h,k) \text{ is dense}\}| > 17 c_2 \log q$.
- Bad events $\mathscr{D}_h$, h is in-sparse. $\mathscr{D}_h$ occurs if $\sum_{k \in X \cap R} [w_{k,h}(1 - z_h) + w_{h,k} z_k] > 0.1$.

LEMMA 6.2. *The bad events described above have probabilities bounded as follows:*

$$\Pr[\mathscr{A}_h] \leq q^{-c_1}, \quad \Pr[\mathscr{B}_T] \leq \prod_{i \in T} p_i,$$
$$\Pr[\mathscr{C}_h] \leq q^{-c_2}, \quad \Pr[\mathscr{D}_h] \leq q^{-c_2}.$$

Proof. The second inequality is trivial. The remaining follow as easy consequences of Theorem C.1. For any machine h, $\mathbf{E}[\sum_{k\in X} w_{h,k}] \le 11\sum_{k\in M} w_{h,k}z_k \le 11$ since $p_i \le 11z_i$ and by Property (B4) with $\theta = 0$. Since $w_{h,k} \le 1$, and c_1, q are large enough, Theorem C.1 implies $\Pr[\sum_{k\in X\cap R} w_{h,k} > c_1 \log q] \le \exp(-c_1 \log q)$.

$\mathbf{E}[\sum_{k\in X\cap R} w_{h,k}] \le 11$ implies for any machine h, the expected number of out-neighbors $k \in X \cap R$ such that (h,k) is dense is at most $11c_2 \log q$. Therefore, the probability that $\mathscr{C}_h$ occurs, by the second inequality of Theorem C.1, is at most $\exp(-(6/11)^2 \cdot 11c_2 \log q/3) \le q^{-c_2}$.

For any machine h, the expected value of $\sum_{k\in X\cap R} w_{k,h}(1-z_h) + w_{h,k}z_k$ is at most $11\delta_0$ since red machines are sampled with probability at most $11z_k \le 11\delta_0$. If q is large enough, $11\delta_0 < 0.005$. Since h is in-sparse, each $w_{k,h} < (c_2 \log q)^{-1}$, and since k is red, $w_{h,k}z_k \le (C_0 \log q)^{-1}$. Therefore $w_{k,h}(1 - z_h) + w_{h,k}z_k \le 2/(c_2 \log q)$. By the first inequality of Theorem C.1, the probability $\mathscr{D}_h$ occurs is at most $\exp(-20 \cdot 0.1 \cdot c_2 \log q/2) = q^{-c_2}$.

LEMMA 6.3. *If none of the bad events occur, then f is $\frac{\delta\delta_0}{2c_1 \log q}$-good for $\delta = (340c_2^2c_3 \log^3 q)^{-1}$ and $\delta_0 = (34c_2c_3 \log q)^{-1}$.*

Proof. We show that if no bad events occur then both conditions of Lemma 6.1 hold. In fact, since $\mathscr{A}_h$ doesn't occur for any h, we get (C1) holds. Fix a subset $T \subseteq X \cap R$. We now prove $\phi(T) + z(T) + \texttt{bnd}(T) \ge \delta|T|$. This is done by careful accounting.

Focus on an in-sparse machine $h \in T$. Since $\mathscr{D}_h$ doesn't occur, we have $\sum_{k\in T}[w_{k,h}(1-z_h) + w_{h,k}z_k] \le 0.1$. By the definition of ϕ_h we have $\phi_h + z_h + \sum_{k\notin T}\left(w_{k,h}(1-z_h) + w_{h,k}z_k\right) \ge 0.9$. Thus, the contribution of h to $\phi(T) + z(T) + \texttt{bnd}(T)$ is at least 0.9.

The red edges induced by $ST \subseteq X \cap R$ form a forest of rooted trees, by the preprocessing step. For each machine in the forest, we ask the root of the tree to contribute to $\phi(T) + z(T) + \texttt{bnd}(T)$. Let h be such a root.

If h is in-sparse, then its contribution is at least 0.9. Otherwise, h is in-dense and we have selected an dense incoming edge (k,h) in the pre-processing step. If k is green, then $k \notin T$. If k is red and out-sparse, then the edge (k,h) is red and thus $k \notin T$. In either case, the contribution of h is at least $(1-z_h)/(c_2 \log q) \ge (1-\delta_0)/(c_2 \log q) \ge 0.99/(c_2 \log q)$.

It remains to consider the case k is red and out-dense, and $k \in T$. In this case, we ask k to contribute to $\phi(T) + z(T) + \texttt{bnd}(T)$. Let $T' = \{k' \in T : (k,k') \text{ is dense}\}$. Since $T \subseteq X \cap R$ and $\mathscr{C}_k$ does not happen, we have $|T'| \le 17c_2 \log q$. $z(T') \le 17c_2\delta_0 \log q$ as all machines in T' are red.

$$\sum_{k'\notin T} w_{k,k'}z'_k \ge \frac{1}{c_2 \log q} \sum_{k'\notin T:(k,k') \text{ dense}} z_\ell \ge \frac{1}{c_2 \log q}\left(\frac{1}{c_3} - 17c_2\delta_0 \log q\right).$$

The quantity is at least $1/(2c_2c_3 \log q)$ since $\delta_0 = (34c_2c_3 \log q)^{-1}$.

We count the number of times each machine is asked to contribute. Since $\mathscr{B}$ events do not happen, every root h is asked at most L times. Since $\mathscr{C}_k$ does not happen, every k is asked by at most $17c_2 \log q$ roots h. Overall, we have proved the lemma for $\delta = \left(2c_2c_3 \log q \cdot L \cdot 17c_2 \log q\right)^{-1} = \left(340c_2^2c_3 \log^3 q\right)^{-1}$.

6.3 Applying the Asymmetric LLL In this section, we show via LLL that no bad event occurs with positive probability. Using the results in [14, 11], we get a polynomial time procedure to obtain an assignment such that no bad events occur. Lemma 6.3 proves Theorem 3.4 for $C = 2^{15}c_1c_2^3c_3^2 \log^5 q$.

We assign each bad event the following $\mathbf{x}$ values: $\mathbf{x}(\mathscr{E}) = 2\Pr[\mathscr{E}]$ for any bad event. The key is arguing about the dependence structure of these events which we undertake next by defining the notion of relevant machines for each event. For any event $\mathscr{A}_h$, the relevant subset of machines is the set $\Gamma(\mathscr{A}_h) = \{k : (h,k) \in G_w\}$ of out-neighbors of h. For any event $\mathscr{B}_T$, the relevant subset of machines is the set $\Gamma(\mathscr{B}_T) = \{k : k \in T\}$. For any event $\mathscr{C}_h$, the relevant subset of machines is the set $\Gamma(\mathscr{C}_h) = \{k : (h,k) \in G_w \text{ and } (h,k) \text{ is heavy}\}$ of heavy out-neighbors of h. For any event $\mathscr{D}_h$, the relevant subset of machines is the set $\Gamma(\mathscr{D}_h) = \{k : (h,k) \in G_w \text{ or } (k,h) \in G_w\}$ of in-neighbors and out-neighbors of h. By the facts that $w_{M,h} \le 1$, and *that all positive z_k and $w_{k,h}$ are at least* $\frac{1}{q}$, we get that $\max_h\{|\Gamma(\mathscr{A}_h)|, |\Gamma(\mathscr{C}_h)|, |\Gamma(\mathscr{D}_h)|\} \le 2q^2$. For a machine $h \in M$, we let $\texttt{group}(h)$ denote the set M_j where $h \in M_j$ if it exists; otherwise $\texttt{group}(h) = \{h\}$.

DEFINITION 6.6. *Two sets S, T of machines are* **group-disjoint** *if no machine in S is in the same group with a machine in T. That is, for any $u \in S, v \in T$, $\texttt{group}(u) \ne \texttt{group}(v)$.*

CLAIM 6.1. *An event $\mathscr{A}_h/\mathscr{B}_T/\mathscr{C}_h/\mathscr{D}_h$ is independent of $\mathscr{A}_k/\mathscr{B}_T/\mathscr{C}_k/\mathscr{D}_k$ if the relevant subset of machines for these events are group-disjoint.*

The main non-trivial lemma is the following.

LEMMA 6.4. *For every red machine $h \in R$, we have $\prod_{S:h\in S}(1 - \mathbf{x}(\mathscr{B}_S)) \ge \exp(-4/3^L)$ where S is over all sets of size L containing h and connected by red edges.*

Proof. To argue about the probability of a connected set S being chosen so that $h \in S$, focus on the red machines $\bigcup_{j \in J_H} M'_j$ and the red edges induced on these machines. The graph is a directed forest. We remove the directions, and focus on the connected component containing h. Root this tree at h and let $\Lambda(v)$ be the set of children of v for any v in this rooted tree. Consider a branching process which selects h with probability $4p_h$ (recall p_h is the probability a machine h gets a heavy job), and if h is chosen, pick each child $k \in \Lambda(v)$ with probability $4p_k$, and continue thus. When this process terminates, we end up with some connected set S. The probability that we get a specific set S is precisely

$$\pi_S := \Pr_{\text{branching process}}[S]$$
$$= (1-4p_h) \prod_{k \in S} \left(\frac{4p_k}{1-4p_k} \prod_{u \in \Lambda(k)} (1-4p_u) \right).$$

Since $p_u \leq 11\delta_0$ as u is red, we have $1 - 4p_u \geq \exp(-5p_u)$. Therefore, $\prod_{u \in \Lambda(k)} (1-4p_u) \geq \exp\left(-5 \sum_{u \in \Lambda(k)} p_u\right)$. If k is out-dense, then k has degree 1 thus has no children; the quantity is 1. If k is out-sparse, then $\sum_{u \in \Lambda(k)} p_u \leq 11 \sum_{u \in \Lambda(k)} z_u \leq 11/c_3$. As $c_3 \geq 200$, we get that $\exp\left(-5 \sum_{u \in \Lambda(k)} p_u\right) \geq 3/4$. This implies $\pi_S \geq \prod_{k \in S} (3p_k) \geq 3^L \Pr[B_S]$. Since $\sum_{S:|S|=L} \pi_S \leq 1$ (the branching process leads to one set S), we get $\sum_{S \ni h} \Pr[B_S] \leq \frac{1}{3^L}$. Now, $\prod_{S:h \in S} (1 - \mathbf{x}(B_S)) \geq \exp(-2 \sum_{S:h \in S} \mathbf{x}(B_S)) = \exp(-4 \sum_{S:h \in S} \Pr[B_S]) \geq \exp(-4/3^L)$.

To check the conditions of the asymmetric LLL is now just book keeping. For any bad event $\mathscr{B}_S$ we have

$$\prod_{T:\mathscr{B}_T \sim \mathscr{B}_S} (1 - \mathbf{x}(\mathscr{B}_T)) \geq \prod_{h \in S} \prod_{k \in \texttt{group}(h)} \prod_{T \ni k} (1 - \mathbf{x}(\mathscr{B}_T))$$
$$\geq \exp(-\frac{4}{3^L}|S||M_j|) \geq \exp(-\frac{4qL}{3^L}),$$

where j is the heavy job which can go to h, and since $z_h \geq 1/q$, we have $|M_j| \leq q$. Since $L = 10 \log q$, the RHS is at least 0.9 whenever $q > 10$.

Fix an event $\mathscr{E}_h$ where $\mathscr{E} \in \{\mathscr{A}, \mathscr{C}, \mathscr{D}\}$. Let us calculate upperbound the product

$$\prod_{S:\mathscr{B}_S \sim \mathscr{E}_h} (1 - \mathbf{x}(\mathscr{B}_S))$$
$$\geq \prod_{k \in \Gamma(\mathscr{E}_h)} \prod_{k' \in \texttt{group}(k)} \prod_{S:k' \in S} (1 - \mathbf{x}(\mathscr{B}_S))$$
$$\geq \exp(-\frac{4}{3^L}|\Gamma(\mathscr{E}_h)||\texttt{group}(k)|) \geq \exp(-\frac{4q^3}{3^L}),$$

where j is the heavy job which contains k' in its M_j. Since $L > 10 \ln q$, the RHS is at least 0.9 since $q > 10$. Similarly,

$$\prod_{h:\mathscr{E}_h \sim \mathscr{B}_S} (1 - \mathbf{x}(\mathscr{E}_h))$$
$$\geq \prod_{k \in S} \prod_{k' \in \texttt{group}(k)} \prod_{h:k' \in \Gamma(\mathscr{E}_h)} (1 - \mathbf{x}(\mathscr{E}_h))$$
$$\geq \exp\left(-2 \sum_{k \in S} \sum_{k' \in \texttt{group}(k)} \sum_{h:k' \in \Gamma(\mathscr{E}_h)} \mathbf{x}(\mathscr{E}_h)\right).$$

The number of terms in the RHS is at most $|S|q_0^3 \leq q_0^4$. Since $\mathbf{x}(\mathscr{E}_h) = 2\Pr[\mathscr{E}_h] \leq q^{-6}$ (if $c_2 \geq 25$ and $c_1 \geq 6$), the RHS is atleast 0.9 for large enough q.

Finally, we have

$$\prod_{k:\mathscr{E}_k \sim \mathscr{E}_h} (1 - \mathbf{x}(\mathscr{E}_k)) \geq \prod_{h' \in S_h} \prod_{h'' \in \texttt{group}(h')} \prod_{k:h'' \in S_k} (1 - \mathbf{x}(\mathscr{C}_k))$$
$$\geq \exp\left(-2 \sum_{h' \in S_h} \sum_{h'' \in \texttt{group}(h')} \sum_{k:h'' \in S_k} \mathbf{x}(\mathscr{C}_k)\right).$$

For any machine h'', the set $\{k : v \in S_k\}$ is precisely the neighbors of h'' (the machines of which h'' is a neighbor of are precisely the neighbors of h''). Therefore, the number of terms in the summation is at most $q^2 \cdot q \cdot q^2 \leq q^5$. Since $\mathbf{x}(\mathscr{C}_k) \leq q^{-6}$, we get that the RHS is at least 0.9 for large enough q. In sum, we get that for any event $\mathscr{E} \in \{\mathscr{A}_h, \mathscr{B}_S, \mathscr{C}_h, \mathscr{D}_h\}$, we have

$$\mathbf{x}(\mathscr{E}) \prod_{T:\mathscr{B}_T \sim \mathscr{E}} (1 - \mathbf{x}(\mathscr{B}_T)) \prod_{k:\mathscr{C}_k \sim \mathscr{E}} (1 - \mathbf{x}(\mathscr{C}_k))$$
$$\times \prod_{k:\mathscr{D}_k \sim \mathscr{E}} (1 - \mathbf{x}(\mathscr{D}_k)) > 0.5\mathbf{x}(\mathscr{E}) = \Pr[\mathscr{E}],$$

implying that x satisfies the LLL condition.

Finally note that the number of events of type $\mathscr{A}, \mathscr{C}, \mathscr{D}$ are polynomially many, and given an assignment of heavy jobs, one can easily check if one of the $\mathscr{B}_T$ occurs or not. Therefore, Theorem C.3 applies and this completes the proof of Theorem 3.4.

References

[1] Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. Combinatorial algorithm for restricted max-min fair allocation. In *Proceedings of the ACM-SIAM Sumposiym on Discrete Algorithms (SODA)*, 2015. 2

[2] Arash Asadpour, Uriel Feige, and Amin Saberi. Santa claus meets hypergraph matchings. *ACM Transactions on Algorithms*, 8(3):24, 2012. 2

[3] Arash Asadpour and Amin Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. *SIAM J. Comput.*, 39(7):2970–2989, 2010. 2

[4] Nikhil Bansal and Maxim Sviridenko. The Santa Claus Problem. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing (STOC)*, 2006. 1

[5] Mohammed-Hossein Bateni, Moses Charikar, and Venkatesan Guruswami. Maxmin allocation via degree lower-bounded arborescences. In *Proceedings of the ACM Symposium on the Theory of Computation (STOC)*, 2009. 2

[6] Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On allocating goods to maximize fairness. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2009. 1, 2

[7] Tomás Ebenlendr, Marek Krcál, and Jiri Sgall. Graph balancing: A special case of scheduling unrelated parallel machines. *Algorithmica*, 68(1):62–80, 2014. 1

[8] Uriel Feige. On allocations that maximize fairness. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 287–293, 2008. 1, 6, 13

[9] Uriel Feige. On estimation algorithms versus approximation algorithms. In *Proceedings of Foundations of Software Technology and Theoretical Computer Science (FSTTCS) conference*, 2008. 0

[10] Uriel Feige and Shlomo Jozeph. Separation between estimation and approximation. *Electornic Colloquium on Computational Complexity (ECCC) Technical Report*, (110), 2014. 0

[11] Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New Constructive Aspects of the Lovász Local Lemma. *J. ACM*, 58(6):28:1–28:28, December 2011. 1, 8, 9, 12, 13, 14

[12] Stavros G. Kolliopoulos and Yannis Moysoglou. The 2-valued case of makespan minimization with assignment constraints. *Information Processing Letters.*, 113(1–2):39–43, 2013. 1

[13] Jan K. Lenstra, David B. Shmoys, and Éva. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46(3):259–271, February 1990. 0, 2, 8

[14] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász Local Lemma. *J. ACM*, 57(2), 2010. 1, 8, 9, 12, 13, 14

[15] Lukas Polacek and Ola Svensson. Quasi-polynomial local search for restricted max-min fair allocation. In *ICALP*, pages 726–737, 2012. 2

[16] Ola Svensson. Santa Claus schedules jobs on unrelated machines. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 617–626, New York, NY, USA, 2011. ACM. 0, 1, 11

[17] José Verschae and Andreas Wiese. On the configuration LP for scheduling on unrelated machines. *J. Scheduling*, 17(4):371–383, 2014. 1

A A $(2-\varepsilon)$ algorithm for the $(1,\varepsilon)$-restricted assignment problem.

THEOREM A.1. *There exists a polynomial time algorithm which returns a $(2-\varepsilon)$-approximation to the makespan minimization problem in $(1,\varepsilon)$-restricted assignment instances. There exists a polynomial time $11/6 \approx 1.833$-factor algorithm to estimate the optimal makespan in $(1,\varepsilon)$-restricted assignment instances.*

Proof. We construct a bipartite matching problem. Vertices on the right side correspond to jobs. Suppose OPT is the optimum makespan. For each machine i, we create $\lfloor \mathsf{OPT} \rfloor$ heavy slots and $\lfloor \mathsf{OPT}/\varepsilon \rfloor$ - $\lfloor \mathsf{OPT} \rfloor$ light slots. There is an edge between a heavy slot and all jobs that can be assigned to the machine; there is an edge between a light slot and all light jobs that can be assigned to the machine. It is easy that there is a matching that covers all jobs. Each machine gets a total load at most $\lfloor \mathsf{OPT} \rfloor + \varepsilon(\lfloor \mathsf{OPT}/\varepsilon \rfloor - \lfloor \mathsf{OPT} \rfloor) = (1-\varepsilon)\lfloor \mathsf{OPT} \rfloor + \varepsilon \lfloor \mathsf{OPT}/\varepsilon \rfloor \leq (2-\varepsilon)\mathsf{OPT}$.

This gives a $2-\varepsilon$ approximation for the problem. By combining this with the $(5/3+\varepsilon)$-estimation algorithm of Svensson [16], we obtain an algorithm that estimates the make span up to a factor of $\min\{2-\varepsilon, 5/3+\varepsilon\} \leq 11/6$.

B Hardness of $(1,\varepsilon)$-restricted assignment problem

We complement our algorithmic result with the following hardness of approximation result.

THEOREM B.1. *For any $\varepsilon > 0$, it is NP-hard to approximate the makespan of the $(1,\varepsilon)$-restricted assignment problem to a factor better than $7/6$.*

Proof. We reduce from the problem of finding a vertex cover in cubic graphs: there exists parameter $K(n)$ so that it is NP hard to decide whether an n-vertex cubic graph has a vertex cover of size $\leq K(n)$ or not. Given a vertex cover instance $G = (V, E)$ on n vertices, we construct an instance of $(P|\gamma|C_{max})$ as follows: we have a machine for every vertex $v \in V(G)$, a set of $n - K(n)$ heavy jobs that can be assigned to any machine, a set of $\frac{1}{3\varepsilon}$ light jobs S_e for every edge $e = (u, v) \in E(G)$ with job $j \in S_e$ having $p_j = \varepsilon$ and can be scheduled on machine u or machine v.

If G has a vertex cover of size $\leq K(n)$, then we can find a schedule of makespan 1. Let $U \subseteq V$ be the vertex cover; allocate all heavy jobs to machines corresponding to $V \setminus U$. For every edge $e = (u, v)$, we are guaranteed one of the end points lies in U and thus doesn't have a heavy job. Allocate all jobs of S_e to that machine. Any machine gets a total small load of at most 1, and any machine getting a heavy job doesn't get a light job.

If G doesn't have a vertex cover of size $\leq K(n)$, then no matter how the heavy jobs are allocated, there must be an edge $e = (u, v)$ such that both u and v are allocated heavy jobs. The total load on one of these two machines is at least $1 + 1/6 = 7/6$.

C Some Useful Tools

We state below two results that we will frequently utilize in our analysis.

THEOREM C.1. *Let Z be the sum of independent scalar random variables each individually in range $[0, K]$ and $\mu = \mathbf{E}[Z]$. Then for any $\lambda \geq 7$, we have*

$$\Pr[Z \geq \lambda\mu] \leq e^{-\lambda\mu/K}.$$

For any $\lambda \in (0, 1)$, we have

$$\Pr[Z \geq (1+\lambda)\mu] \leq e^{-\lambda^2\mu/3K},$$
$$\Pr[Z \leq (1-\lambda)\mu] \leq e^{-\lambda^2\mu/2K}.$$

Proof. All those bounds are simple application of standard Chernoff bounds. Let X be the sum of n independent random variables, each take value in $[0, 1]$. Let $\mu = \mathbb{E}[X]$. Then for every $\delta > 0$, we have

$$\Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}.$$

For every $\delta \in (0, 1)$, we have

$$\Pr[X \leq (1-\delta)\mu] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu}.$$

To prove the theorem, we can scale the random variables by a factor of $1/K$ and then mean of Z is changed to μ/K. Thus, we can assume $K = 1$.

The first inequality is obtained by setting $\delta = \lambda - 1$ and observing that $e^{\lambda-1}/\lambda^{\lambda} \leq e^{-\lambda}$ for $\lambda \geq 7$. For the second inequality and third inequality, let $\delta = \lambda$. The second inequality holds since $e^{\delta}/(1+\delta)^{1+\delta} = \exp\big(\delta - (1+\delta)\ln(1+\delta)\big) \leq \exp\big(\delta - (1+\delta)\frac{2\delta}{2+\delta}\big) = \exp\left(\frac{-\delta^2}{2+\delta}\right) \leq \exp(-\delta^2/3)$. The third inequality holds since $e^{-\delta}/(1-\delta)^{1-\delta} = \exp\big(-\delta - (1-\delta)\ln(1-\delta)\big) \leq \exp\big(-\delta - (1-\delta)(-\frac{2\delta}{2-\delta})\big) = \exp\left(-\frac{\delta^2}{2-\delta}\right) \leq \exp(-\delta^2/2)$.

THEOREM C.2. (ASYMMETRIC LLL) *Let $\mathcal{E} = \{E_1, \ldots, E_m\}$ be a finite collection of (bad) events in a probability space. For each E_i, let $\Gamma(E_i)$ denote a subset of events such that E_i is independent of each event in $\mathcal{E} \setminus (E_i \cup \Gamma(E_i))$. Then if there exists an assignment $\mathbf{x} : \mathcal{E} \to (0, 1)$ satisfying the property $\Pr[E_i] \leq \mathbf{x}(E_i) \cdot \prod_{E_j \in \Gamma(E_i)} (1 - \mathbf{x}(E_j))$, the probability that none of the events in $\mathcal{E}$ occurs is at least $\prod_i (1 - \mathbf{x}(E_i))$.*

While the version stated above is only an existence statement, the recent work of Moser and Tardos [14], and Haeupler et al. [11] has given polynomial-time algorithms for finding a solution that avoids all bad events. We use the notation $E_j \sim E_i$ to indicate that $E_j \in \Gamma(E_i)$. Let $V = \{v_1, v_2, \cdots, v_n\}$ be n independent random variables. Let $\mathcal{E} = \{E_1, E_2, \cdots, E_m\}$ be a finite collection of (bad) events where each E_i only depends on a subset $V_i \subseteq V$ of variables. Let $\Gamma(E_i) = \{E_{i'} : i' \neq i, V_i \cap V_{i'} \neq \emptyset\}$.

The Moser-Tardos (MT, henceforth) algorithm does the following: a) Initially sample v_i's independently, and b) until all E_i's are dissatisfied, pick an arbitrary satisfied E_i and resample the v_j's present in V_i. Moser and Tardos [14] showed that if the LLL condition held, the above algorithm terminated in $O\left(\sum_{i=1}^{m} \mathbf{x}(E_i)(1 - \mathbf{x}(E_i))^{-1}\right)$ iterations. This suffices for many applications; however there are two issues – a) m could be superpolynomial in n, and b) given a setting of v_i's there may not be an efficient method to detect if a satisfied E_i exists or not. Haeupler et al. [11] addressed these issues in the following ways.

THEOREM C.3. *(Paraphrasing of Theorem 3.1 in [11]) Suppose the LLL condition holds, and let $\delta := \min_j \mathbf{x}(E_i) \prod_{j \sim i}(1 - \mathbf{x}(E_j))$. Then the expected number of resamplings of the MT algorithm is at most $n \log(1/\delta) \max_i (1 - \mathbf{x}(E_i))^{-1}$.*

The above theorem takes care of situations where the number of events may be superpolynomial; however, 'efficient verifiability' occurs, that is, given a setting of v_i's one can detect a satisfied E_j or assert none hold. To take care of issue (b) above, Haeupler et al. [11] modified the MT algorithm as follows. It parametrizes the events with a set $\mathcal{E}' \subseteq \mathcal{E}$ of *core events*. Randomly and independently assign a value to each random variable in V. In each iteration, we check if any bad event $E_i \in \mathcal{E}'$ happens. If there is such a bad event $E_i \in \mathcal{E}'$, we resample all variables in V_i and start a new iteration. Otherwise we terminate the algorithm return the current assignment.

THEOREM C.4. *(Paraphrasing of Theorem 3.4 in [11]) Suppose there exists an $\varepsilon \in (0, 1)$ and assignment $\mathbf{x} : \mathcal{E} \mapsto (0, 1 - \varepsilon)$ such that a slightly stronger-than-LLL condition holds:*

$$\Pr^{1-\varepsilon}[E_i] \leq \mathbf{x}(E_i) \cdot \prod_{E_j \in \Gamma(E_i)} (1 - \mathbf{x}(E_j)) \tag{C.1}$$

Suppose further, that $\log(1/\delta) \leq \mathrm{poly}(n)$. Then

1. *For any $p \geq 1/\mathrm{poly}(n)$, the set $\mathcal{E}' := \{E_i : \Pr[E_i] \geq p\}$ is of size at most $\mathrm{poly}(n)$.*

2. *With probability* $(1 - n^{-c})$*, the HSS algorithm with core events* $\mathcal{E}'$ *terminates after* $O(n \log n)$ *resamplings and returns an assignment such that no event in* $\mathcal{E}$ *occurs.*

D Omitted details from §5

CLAIM 5.1. *If* (S, T) *is a* δ*-witness, then there is a connected* δ*-witness* $(\tilde{S}, \tilde{T})$*, with* $\tilde{S} \subseteq S$ *and* $\tilde{T} \subseteq T$*.*

Proof. Consider all the (weakly) connected components of $G_w[S]$(the sub-graph of G_w induced by S). There must be some connected component induced by $\tilde{S} \subseteq S$ such that $\left|T \cap \tilde{S}\right| + w_{\tilde{S},\tilde{S}} > (2-\delta)\left|\tilde{S}\right|$, since summing up the left side over all connected components $\tilde{S}$ gives $|T|+w_{S,S}$ and summing up the right side gives $(2-\delta)|S|$. Thus, $\left(\tilde{S}, \tilde{T} = T \cap \tilde{S}\right)$ is a connected δ-witness.

Claim 5.2 follows immediately from Theorem 3.1 and Claim 5.1.

D.1 Proof of Lemma 5.1 Before the proof of the Lemma, we need one simple claim.

CLAIM D.1. *For any* (p, q, θ)*-canonical instance, we have* $w_{M,h} \leq 1.1, w_{h,M} \leq 1.1p$ *for every* $h \in M$*.*

Proof. Since $z_h + w_{M,h}(1 - z_h) \leq 1 + \theta \leq 1.05$ and $z_h \leq 1/2$, we have $w_{M,h} \leq \frac{1.05 - z_h}{1 - z_h} \leq 1.1$.

Consider any machine $h \in M$. Notice that $\sum_{k \in M} w_{h,k} z_k \leq 1 + \theta \leq 1.1$ and $z_k \geq 1/p$ if $w_{h,k} > 0$. We have $w_{h,M} \leq \frac{1.1}{(1/p)} = 1.1p$.

LEMMA 5.1. *Let* $\mathsf{I} = (\{M_j : j \in J_{\mathsf{H}}\}, w, z)$ *be a* (p, q, θ)*-canonical instance. Assume* $q \geq \max\{p, q_0\}$*. Then, we can find in polynomial time a* (p, q', θ')*-canonical instance* $\mathsf{I}' = (\{M_j : j \in J_{\mathsf{H}}\}, w', z)$*, such that any* δ'*-good assignment* f *for* I' *is* δ*-good for* I*, where* $q' = q/2, \theta' = \theta + 8\sqrt{\log q/q}, \delta' = \delta + 8\sqrt{\log q/q}$*.*

Proof. For each pair (h, k) with $0 < w_{h,k} < 1/q' = 2/q$, we let $w'_{h,k} = 1/q'$ with probability $q' w_{h,k}$ and let $w'_{h,k} = 0$ with probability $1 - q' w_{h,k}$. For all other pairs (h, k), we let $w'_{h,k} = w_{h,k}$. Then $\mathsf{I}' = (\{M_j : j \in J_{\mathsf{H}}\}, w', z)$ is the new canonical instance.

1. $\mathscr{A}_h$, for every machine $h \in M$: $\mathscr{A}_h$ occurs if $z_h + w'_{M,h}(1 - z_h) + \sum_{k \in M} w'_{h,k} z_k > 1 + \theta'$;
2. $\mathscr{B}_{S,T}$, for every connected δ-witness (S, T) of I: $\mathscr{B}_{S,T}$ occurs if (S, T) is not a δ'-witness of I'.

If none of the bad events occur, then I' is a $(p, q/2, \theta')$-canonical instance; furthermore, any δ'-good assignment for I' must be a δ-good assignment for I since otherwise $\mathscr{B}_{S,T}$ would occur for some connected δ-witness. In the rest of the proof, we use LLL to show that none of the bad events occur with positive probability. Using the techniques of [14, 11], there is a polynomial time procedure which obtains I' with the desired property.

Focus on the quantity W on the left side of the inequality defining $\mathscr{A}_h$. All random variables (the $w'_{h,k}$s) in W take value in $\{0, 1/q'\}$ and the coefficient before each random variable in Z is at most 1; moreover, $\mathbb{E}(w'_{h,k}) = w_{h,k}$. By Property (B4), we have $z_h + w_{M,h}(1 - z_h) + \sum_{k \in M} w_{h,k} z_k \leq 1 + \theta < 1.1$. The Chernoff bound in Theorem C.1 gives that the probability that $\mathscr{A}_h$ occurs is at most $\exp\left(-(\theta' - \theta)^2 q'/3.3\right) \leq \exp(-8 \log q) = q^{-8}$.

Now consider the bad event $\mathscr{B}_{S,T}$. Since (S, T) is a δ-witness of I, we have $|T| + w_{S,S} > (2 - \delta)|S|$. $\mathscr{B}_{S,T}$ occurs if $|T| + w'_{S,S} \leq (2 - \delta')|S|$. Again, by Chernoff bound, the probability that $\mathscr{B}_{S,T}$ occurs is at most $\exp\left(-(\delta' - \delta)^2 q' |S|/4\right) \leq e^{-8(\log q)|S|} = q^{-8|S|}$.

Now we apply the (asymmetric) LLL. In order to apply LLL, we need to define the $\mathbf{x}$ values for the bad events. Define $\mathbf{x}(\mathscr{A}_h) = q^{-7}$ and $\mathbf{x}(\mathscr{B}_{S,T}) = q^{-7|S|}$.

Focus on some bad event $\mathscr{A}_h$. If $\mathscr{A}_k$ is dependent of $\mathscr{A}_h$, then either $w_{k,h} > 0$, or $w_{h,k} > 0$ and $z_k > 0$. By Claim D.1, the number of events $\mathscr{A}_k$ dependent of $\mathscr{A}_h$ is at most $(1.1 + 1.1p)/(1/q) < q^3/4$ since each positive $w_{h,k}$ has $w_{h,k} \geq 1/q$ and $p \leq q$. We count the number of events $\mathscr{B}_{S,T}$ dependent on $\mathscr{A}_h$ satisfying $|S| = t$. $\mathscr{B}_{S,T}$ is dependent on $\mathscr{A}_h$ only if $h \in S$. Since the degree of vertices in G_w is at most $q^3/4$, and $G_w[S]$ is connected, the number of sets S is at most $(q^3)^t$. [2] For a fixed S, there are at most 2^t different sets T. Thus, the number of such dependent events is at most $(q^3)^t \times 2^t \leq q^{4t}$. This gives,

$$\mathbf{x}(\mathscr{A}_h) \prod_{\mathscr{E} \sim \mathscr{A}_h} (1 - \mathbf{x}(\mathscr{E})) \geq q^{-7}\left(1 - q^{-7}\right)^{q^3} \prod_{t \geq 1}\left(1 - q^{-7t}\right)^{q^{4t}} \geq q^{-8} \geq \Pr(\mathscr{A}_h),$$

where the product in the LHS is over all events $\mathscr{E}$ dependent on $\mathscr{A}_h$.

Now consider some bad event $\mathscr{B}_{S,T}$ with $|S| = s$. Using a similar counting argument, the number of events $\mathscr{A}_h$ that are dependent on $\mathscr{B}_{S,T}$ is at most sq^3

[2] We can use the same argument as in [8]. Given a graph G of degree d and a vertex v, we want to bound the number of induced connected sub-graphs of s containing v. Fix an arbitrary spanning tree for the sub-graph and root it at v. There are at most $2^{2s} = 4^s$ tree structures: visiting the tree in the DFS order and we only need to specify which $s - 1$ edges are forward moves. Given a tree structure, there at at most d^s choices for the tree. Thus the number is bounded by $(4d)^s$.

and the number of events $\mathscr{B}_{\tilde{S},\tilde{T}}$ dependent on $\mathscr{B}_{S,T}$ satisfying $\tilde{S} = t$ is at most sq^{4t}. Thus,

$$\begin{aligned}
&\mathbf{x}\left(\mathscr{B}_{S,T}\right) \prod_{\mathscr{E}\sim\mathscr{B}_{S,T}} \left(1-\mathbf{x}(\mathscr{E})\right) \\
&\geq q^{-7s}\left(1-q^{-7}\right)^{sq^3} \prod_{t\geq 1}\left(1-q^{-7t}\right)^{sq^{4t}} \\
&\geq q^{-8s} \geq \Pr(\mathscr{B}_{S,T}).
\end{aligned}$$

We have verified that the conditions for the asymmetric LLL, and this completes the proof of the lemma.

To see how the theorems of [14, 11] can be applied, note that that there are at most m events of the type $\mathscr{A}_h$. The events $\mathscr{B}_{S,T}$ are exponentially many and do not seem to be efficiently verifiable. This is where one uses Theorem C.4 of [11]. Note in the above analysis, (C.1) holds with $\varepsilon = 1/7$. The theorem implies the 'core bad events' which have $\Pr[\mathscr{B}_{S,T}] \geq 1/\mathrm{poly}(m)$, that is, those with $|S| = O(\frac{\log m}{\log q})$, are at most $m^{O(1)}$. Since these can be enumerated over using a BFS tree, we can find a 'good' assignment in polynomial time.

D.2 Proof of Lemma 5.2

LEMMA 5.2. *Let $\mathsf{I} = (\{M_j : j \in J_{\mathsf{H}}\}, w, z)$ be a (p,q,θ)-canonical instance, where $p \geq \max\{q, q_0\}$. We can find in polynomial time a (p',q,θ')-canonical instance $\mathsf{I}' = (\{M_j : j \in J_{\mathsf{H}}\}, w, z')$ such that any δ-good solution f for I' is also δ-good for I, where $p' = p/2, \theta' = \theta + 8\sqrt{\log p/p}$.*

Proof. For every $h \in M$ with $0 < z_h < 1/p' = 2/p$, we let $z'_h = 1/p'$ with probability $p'z_h$ and let $z'_h = 0$ with probability $1 - p'z_h$. For all other machines h, we let $z'_h = z_h$. Note that $\mathbf{E}[z'_h] = z_h$.

To make $(\{M_j : J \in J_{\mathsf{H}}\}, w, z')$ a canonical instance, we need to apply more operations. If some $h \in M_j$ has $z'_h = 0$, we need to remove h from M_j. If $z'_k = 0$, for every $h \neq k$, we need to change the $w_{h,k}$ light load of type-(h,k) to load of type-(k,k). However, these operations do not affect our proof. Thus, we can pretend our new instance is $\mathsf{I}' = (\{M_j : J \in J_{\mathsf{H}}\}, w, z')$.

Since the definition of a δ-good assignment is independent of z, a δ-good assignment for I' is a δ-good assignment for I. The non-trivial part is to show that I' is (p',q,θ')-canonical. The non-trivial properties are (B3) and (B4). Note that (B1) is satisfied by the construction above and (B2) is untouched.

To this end, consider the following two types of bad events. If none of the bad events occur we are done. Once again, we use LLL to show that none of the bad events occur with positive probability, and the lemma is proven by the theorems of [14, 11].

1. $\mathscr{A}_h, h \in M$: $\mathscr{A}_h$ occurs if $z'_h + w_{M,h}(1 - z'_h) + \sum_{k\in M} w_{h,k} z'_k > 1 + \theta'$.
2. $\mathscr{B}_j, j \in J_{\mathsf{H}}$. $\mathscr{B}_j$ occurs if $\sum_{h\in M_j} z'_h < 0.2 - \theta'$.

Consider the quantity Z on the left side of the inequality defining $\mathscr{A}_h$. Notice that we have $z_h + w_{M,h}(1-z_h) + \sum_{k\in M} w_{h,k} z_k \leq 1 + \theta < 1.1$. All random variables z'_k take value in $\{0, 1/p'\}$; moreover, we have $\mathbb{E}[z'_k] = z_k$. The coefficient before each $z'_k, k \neq h$ in Z is at most $w_{h,k} \leq 1.1$. The coefficient before z'_h is at most 1 but might as small as -0.1. If z'_h is not fixed and the coefficient before it is negative, we define $y = 1/p' - z'_h$ and replace the random variable z'_h with y. The Chernoff bound in Theorem C.1 gives that $\mathscr{A}_h$ happens with probability $\exp\left(-(\theta' - \theta)^2 p'/4\right) = \exp\left(-(\theta' - \theta)^2 p/8\right) = e^{-8\log p} = p^{-8}$.

Now focus on $\mathscr{B}_j$ for some $j \in J_{\mathsf{H}}$. Since the unfixed random variable z'_h takes value between 0 and $1/p' = 2/p$, the Chernoff bound gives that the probability that $\mathscr{B}_j$ occurs is at most $\exp\left(-(\theta' - \theta)^2 p/8\right) = p^{-8}$. In order to apply the uniform LLL, we need to upper-bound the number of bad events that each $\mathscr{A}_h$ (or $\mathscr{B}_j$) depends on. $\mathscr{A}_h$ and $\mathscr{A}_k$ are dependent only if h and k are adjacent G_w; $\mathscr{A}_h$ and $\mathscr{B}_j$ are dependent if there exists $k \in M_j$ such that h and k are adjacent in G_w. Since each $|\{k \in M_j : z_k > 0\}| \leq p$ and since the degree of the graph G_w is at most $2qp + 4q \leq 3p^2$ by Claim D.1, any bad event is dependent on at most $(3p^2)p \leq p^8/e$ other events. Thus the symmetric LLL conditions hold, and thus with positive probability none of the bad events occur. The polynomial time algorithm follows directly from [14] since the number of bad events is polynomially many.