On Minimizing the Makespan When Some Jobs Cannot Be Assigned on the Same Machine[*]

Syamantak Das[1] and Andreas Wiese[2]

1 **University of Bremen, Bremen, Germany**
`syamanta@uni-bremen.de`

2 **Department of Industrial Engineering and Center for Mathematical Modeling, Universidad de Chile, Chile**
`awiese@dii.uchile.cl`

Abstract

We study the classical scheduling problem of assigning jobs to machines in order to minimize the makespan. It is well-studied and admits an EPTAS on identical machines and a $(2-1/m)$-approximation algorithm on unrelated machines. In this paper we study a variation in which the input jobs are partitioned into *bags* and no two jobs from the same bag are allowed to be assigned on the same machine. Such a constraint can easily arise, e.g., due to system stability and redundancy considerations. Unfortunately, as we demonstrate in this paper, the techniques of the above results break down in the presence of these additional constraints.

Our first result is a PTAS for the case of identical machines. It enhances the methods from the known (E)PTASs by a finer classification of the input jobs and careful argumentations why a good schedule exists after enumerating over the large jobs. For unrelated machines, we prove that there can be no $(\log n)^{1/4-\epsilon}$-approximation algorithm for the problem for any $\epsilon > 0$, assuming that $\mathsf{NP} \nsubseteq \mathsf{ZPTIME}(2^{(\log n)^{O(1)}})$. This holds even in the restricted assignment setting. However, we identify a special case of the latter in which we can do better: if the same set of machines we give an 8-approximation algorithm. It is based on rounding the LP-relaxation of the problem in phases and adjusting the residual fractional solution after each phase to order to respect the bag constraints.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases approximation algorithms, scheduling, makespan minimization

Digital Object Identifier 10.4230/LIPIcs.ESA.2017.31

1 Introduction

Minimizing the makespan is a classical problem in scheduling [8, 9]. Given a set of machines M and set of jobs J, we seek to assign each job to a machine. In the setting where all machines are *identical*, the processing time of each job j is given by a value p_j for each job j. For *unrelated* machines the processing time of a job j can depend on the machine i on which it is scheduled. In this case the input contains a value $p_{ij} \in \mathbb{R}_0^+ \cup \{\infty\}$ for each combination of a machine i and a job j. The objective is to minimize the makespan, i.e., the maximum load of a machine i which is the total processing time of jobs assigned to i. The problem is well-studied, for identical machines it is strongly NP-hard and there are PTASs [11, 17] and even EPTASs, e.g., [12, 13, 10]. For unrelated machines there is a 2-approximation algorithm

[*] This work was partially supported by the Millennium Nucleus Information and Coordination in Networks ICM/FIC RC130003.

due to Lenstra, Shmoys, and Tardos [16], an improvement to $2 - 1/m$ due to Shchepin and Vakhania [19], and a lower bound of 3/2 [16].

In practice, one often finds side constraints in addition to the above scheduling setting that make the problem harder. A typical constraint is that some jobs have to be assigned on different machines. For instance, on-board computers of aeroplanes typically have several CPUs (modeled as machines) and for system stability considerations some tasks need to be executed on different CPUs [6]. The idea is that if one CPU fails then the plane still continues to operate safely. Minimizing the makespan is closely related to the bin packing problem where the bins and the items correspond to the machines and the jobs, respectively. There are several applications of bin packing where the items are partitioned into groups and no two items from the same group can be assigned to the same bin, for instance in distributed systems and other settings, see [18].

To model the above, in this paper we assume that the input jobs are partitioned into *bags* $J = B_1 \dot\cup B_2 \dot\cup ... \dot\cup B_b$ and that no two jobs from the same bags are allowed to be assigned on the same machine. We call these new requirements the *bag-constraints*.

In this paper we study the problem of minimizing the makespan on identical and unrelated machines with bag-constraints.

1.1 Identical machines

The known (E)PTASs [12, 13, 10, 11, 17] for minimizing the makespan on identical machines follow the idea of enumerating the solution for the large jobs, e.g., jobs that are larger than $\epsilon \cdot OPT$, and then adding the small jobs via a greedy algorithm. More precisely, one enumerates patterns for the large jobs that indicate how many large jobs of each size are assigned on each machine. The large jobs are then assigned according to these patterns and it does not matter which exact job is assigned to which slot of each pattern as long as the size of the slot is respected. In the case of bag-constraints this unfortunately does not work directly anymore. One can still enumerate the mentioned patterns and, with some additional effort, assign the large jobs to them such that they respect the bag-constraints. However, we cannot guarantee that the large jobs are assigned exactly like in the optimal solution. It could be that the jobs from the different bags are distributed completely differently on the machines than in the optimal solution (while still respecting the enumerated slots). In fact, there are instances for which the above procedure can lead to an assignment of the large jobs such that *any* solution for the remaining jobs has a makespan of at least $(2 - O(\epsilon))OPT$, see Figure 1 for an example.

Hence, we need additional ideas for the setting with bag-constraints. First, we observe that in the mentioned example many bags have relatively many large jobs (more than $\epsilon \cdot m$ many). There can be only $O_\epsilon(1)$ such *large bags* and hence we can afford to be more careful for them when we enumerate their large jobs. Indeed, we manage to assign the large jobs in such bags as in an optimal solution. Then we assign all other large jobs according to the enumerated pattern such that we respect the bag-constraints. To assign the remaining (non-large) jobs, we partition them into medium and small jobs such that the total processing time of the medium jobs is small, at most $\epsilon^2 OPT \cdot m$. We find a way to assign the latter to the machines such that via some swapping and charging arguments we can guarantee that for the remaining small jobs there exists a solution with small overall makespan. For the small jobs the argumentation is again not as easy as without the bag-constraints since some machines already have jobs from some bags which prevents small jobs of such bags to be assigned to them. We solve the remaining problem with a combination of a dynamic programming algorithm and a modified greedy routine. Overall, we obtain a PTAS for minimizing the makespan under bag-constraints. Hence, like without bag-constaints, there is

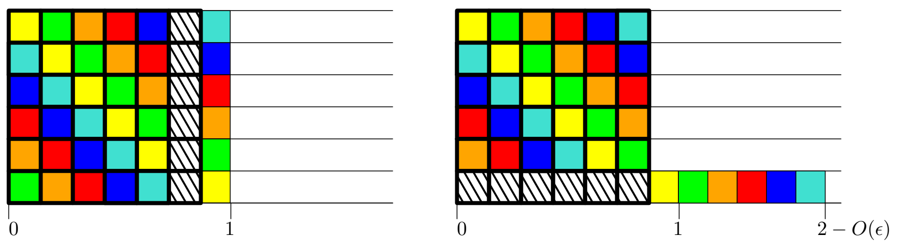


Figure 1 Left: an optimal schedule for the given instance. The bold lines indicate the enumerated patterns for the big jobs, all of them having size ϵ. The colors show the different bags of the jobs. Each white (striped) job j is in a (private) bag that contains only j. Right: a schedule in which the big jobs are assigned according to the same patterns but differently than in OPT. Thus, all non-big jobs have to be assigned to the last machine in order to satisfy the bag-constraints. This yields an approximation ratio of $2 - O(\epsilon)$.

a $(1+\epsilon)$-approximation in polynomial time, but clearly new ideas are necessary to construct such an algorithm.

▶ **Theorem 1.** *There is a PTAS for minimizing the makespan on identical machine with bag-constraints.*

1.2 Unrelated machines

For makespan minimization on unrelated machines, the mentioned LP-based 2- and $(2-1/m)$-approximation algorithms [16, 19] are known. There are several rounding strategies for the natural LP such as an argumentation via bipartite matchings [16], rounding via (sparse) extreme point solutions [19], and, related to the latter, iterated rounding [20]. The bag-constraints induce a linear constraint for each combination of a bag and a machine. Thus, it seems natural to enhance the normal LP by these constraints and try to adapt one of the known rounding techniques. However, in this paper we show that this is deemed to fail. We prove that on unrelated machines the problem is hard to approximate with a ratio of $(\log n)^{1/4-\epsilon}$ for any $\epsilon > 0$. This holds also in the restricted assignment case where each job j has a size p_j and there are some machines on which it cannot be assigned, i.e, $p_{ij} \in \{p_j, \infty\}$ for each job j and each machine i. On the other hand, we show that a randomized rounding algorithm yields a $O(\log n/\log\log n)$-approximation.

▶ **Theorem 2.** *For minimizing the makespan on unrelated machines with bag-constraints there can be no $(\log n)^{1/4-\epsilon}$-approximation algorithm for any $\epsilon > 0$ unless* $\mathsf{NP} \subseteq \mathsf{ZPTIME}(2^{(\log n)^{O(1)}})$. *This holds even for the restricted assignment case.*

Thus, in contrast to the case of identical machines we see here an increase in complexity due to the bag-constraints. However, we identify a special case of the restricted assignment setting where we can do better than in the general case: if all jobs from each bag can be assigned to exactly the same set of machines then we obtain a 8-approximation algorithm based on the above mentioned LP. For this we need several new ideas on top of the above mentioned known rounding techniques. First, we round the job sizes to powers of 2 and process the jobs in groups according to their sizes. We show that if all jobs have exactly the same size then even with the bag-constraints the LP is almost exact. We then assign the jobs with largest size via LP-rounding. Together with the fractional solution of the remaining jobs this might violate the bag-constraints. However, by carefully exploiting the properties of our special case we are able to construct a new fractional solution for the remaining jobs that satisfies the bag-constraints. We continue iteratively. When we change the residual fractional solution after each iteration we employ some careful geometric sum arguments in order to ensure that the final solution has a makespan that is at most by a factor 8 larger than the value of the initial LP-solution.

▶ **Theorem 3.** *There is a 8-approximation algorithm for minimizing the makespan with bag-constraints in the restricted assignment case if for each bag all jobs in the bag can be assigned to the same set of machines.*

Due to space constraints we give the statement of several lemmas and theorems without proofs and details in this extended abstract.

1.3 Other related work

Makespan. For the restricted assignment case without bag-constraints, Svensson [21] gave an estimation algorithm with a ratio of $33/17+\epsilon$, i.e., his algorithm can estimate the optimal makespan up to this factor in polynomial time but does not necessarily find the corresponding schedule within this time bound. This algorithm was improved recently by Jansen and Rohwedder [15] to an $(11/6+\epsilon)$-estimation algorithm. If there are only two different jobs sizes, there is even a 5/3-estimation due to Jansen, Land, and Maack [14]. Moreover, for the special case that each job has either size 1 or size ϵ there is a $(2-\delta)$-approximation algorithm due to Chakrabarty, Khanna, and Li for a small constant $\delta > 0$ [2]. In contrast to the previous algorithms, it computes the actual schedule in polynomial time. All above algorithms are based on the configuration-LP in the restricted assignment case. Note that for general unrelated machines the latter LP has an integrality gap of (asympotically) 2 [5, 22].

Scheduling with conflicts. Typically, in these settings, there is an underlying conflict graph with the jobs forming the vertices; there is an edge between any two vertices if and only if the corresponding jobs cannot be scheduled on the same machine. In [1], for example, the authors give a tight 2-approximation algorithm for makespan minimization when the underlying conflict graph is polynomial time colorable. A slightly different setting is considered in [7]. Here, an edge between any two jobs in the conflict graph dictates that they cannot be scheduled in overlapping intervals on different machines. The authors, among other results, prove that the makespan minimization problem is APX-hard even for 4 different job sizes and give 4/3-approximation algorithms for the case of three different job sizes and an exact algorithm for two different job sizes. A related setting, called the multi-level bottleneck assignment is considered in [4]. It can be thought of as a generalization of our setting where each bag has the same number of jobs and the additional restriction that each machine gets exactly the same number of jobs in a schedule. The authors prove a 2-approximation for the special case with 3 bags.

2 A PTAS for identical machines

In this section we present our PTAS for minimizing the makespan under bag-constraints on an arbitrary number of identical machines. As we will see, the standard techniques of enumerating over large jobs and then adding small jobs greedily are not sufficient since a bag can contain large and small jobs and, therefore, these jobs interact with each other much more than without the bag-constraints.

Let $\epsilon > 0$ and assume for simplicity that $1/\epsilon \in \mathbb{N}$. First, we assume that we guess the optimal makespan $T^\star$ via a binary search framework and we assume by scaling that $T^\star = 1$.

We round all job lengths to powers of $1+\epsilon$, i.e., we assume that for each job $j \in J$ we have that $p_j = (1+\epsilon)^k$ for some $k \in \mathbb{N}$. Due to this we lose at most a factor of $1+\epsilon$ in the objective.

2.1 Straight-forward approach

As mentioned in the introduction, a natural approach would be to classify jobs into large and small jobs, e.g., define a job j to be large if $p_j \geq \epsilon$ and small otherwise and to enumerate over the large jobs. More precisely, one would enumerate over the patterns of the large jobs where a pattern indicates how many large jobs of each size are assigned to a machine. Since each machine can have at most $1/\epsilon$ large jobs and there are only $O(\log_{1+\epsilon} 1/\epsilon)$ many different sizes of large jobs, this yields $(1/\epsilon)^{O(\log_{1+\epsilon} 1/\epsilon)} =: K_\epsilon$ many different patterns. Thus, in time $(m+1)^{K_\epsilon}$ we can enumerate how many machines follow each pattern and thus enumerate the machine patterns of the optimal solution (up to permutation of machines). Then, one would compute a solution for the large jobs following the enumerated patterns, e.g., via assigning them greedily to the patterns' slots. However, the computed assignment of jobs to slots might be different than in the optimal solution and if a large job is assigned to a machine i then a small job from the same bag cannot be assigned to i anymore. Figure 1 shows an example where such an algorithm enumerates the patterns of some optimal solution but then assigns the large jobs differently than OPT such that any assignment for the remaining small jobs yields a makespan of at least $(2-O(\epsilon))T^\star$. Hence, this approach does not work directly.

2.2 Refined job classification and enumeration

Instead, we use a classification of the jobs into large, medium, and small jobs. Using a standard shifting argument we define these groups such that the medium jobs have small total processing time.

▶ **Lemma 4.** *For any given instance we can compute a value $k \in \{1, \ldots, 1/\epsilon^2\}$ such that $\sum_{j \in J: p_j \in [\epsilon^{k+1}, \epsilon^k)} p_j \leq m \cdot \epsilon^2$.*

With the value k from Lemma 4 we define a job j to be *large* if $p_j \geq \epsilon^k$, *medium* if $p_j \in [\epsilon^{k+1}, \epsilon^k)$ and *small* if $p_j < \epsilon^{k+1}$. Note that in the example in Figure 1 there are some bags that have a large number of large jobs ($m-1$ many). We call a bag *large* if it contains at least $\epsilon \cdot m$ large or medium jobs and *small* otherwise. The following proposition shows that there can be only constantly many large bags (since otherwise the total processing time of their jobs would be bigger than m).

▶ **Proposition 5.** *There can be at most $O(1/\epsilon^{k+2})$ large bags.*

In our algorithm, we want to enumerate over patterns that contain large jobs and additionally medium jobs from large bags. However, for the large and medium jobs in large bags we want to be more careful: we want to assign them to the slots of the enumerated pattern like in an optimal solution. Since there are only $O_\epsilon(1)$ large bags, we can incorporate the enumeration of this this assignment in to the enumeration of the patterns.

Assume that after our rounding the medium and large jobs have sizes $\mathcal{S} = \{s_1, \ldots, s_{|\mathcal{S}|}\}$ with $|\mathcal{S}| = O(\log_{1+\epsilon}(1/\epsilon^{k+1}))$. A pattern p consists of at most $(1+\epsilon)/\epsilon^{k+1}$ *slots* (note that each machine can have at most $(1+\epsilon)/\epsilon^{k+1}$ jobs that are medium or large) where each slot is characterized by a size $s \in \mathcal{S}$ and a label that specifies either one of the $O(1/\epsilon^{k+2})$ large bags (and then only jobs from that bag can be assigned to the slot) or that the slot can be used only for jobs from small bags. If s belongs to the size of a medium job then we do not allow the latter type of label, i.e., we allow slots of medium size only for jobs from large bags. Let $K'_\epsilon = O_\epsilon(1)$ be the total number of patterns. Hence, in time $(m+1)^{K'_\epsilon}$ we can guess a pattern for each machine such that all patterns together correspond to an optimal solution. From these patterns we can directly conclude the complete assignment of medium and large jobs from the large bags. Hence, we obtain an assignment for the latter jobs and a pattern for the large jobs in small bags.

▶ **Lemma 6.** *In time m^K, where $K = (\frac{1}{\epsilon^2}\log(\frac{1}{\epsilon}))^{O(\frac{1}{\epsilon^3})}$ we can guess the assignment of the large and medium jobs from the large bags and a pattern for each machine for the large jobs in small bags (both corresponding to an optimal solution).*

Next, we assign the large jobs of the small bags to the slots given by the enumerated pattern. We do this via a dynamic program (DP). This DP will successfully assign all remaining large jobs, however, not necessarily to the slots to which the optimal solution assigned them.

▶ **Lemma 7.** *There is a dynamic program that assigns all large jobs in small bags to the machines such that (i) no two large jobs from the same (small) bag are assigned to the same machine and (ii) for each size $s \in \mathcal{S}$ each machine i gets the same number of jobs of size s as there are slots of size s for jobs from small bags in the pattern assigned to i.*

2.3 Assignment of remaining medium jobs

So far we have assigned all large jobs and additionally all medium jobs in large bags. We want to assign the medium jobs from the small bags now. If we were allowed to assign each such job to any machine then we could distribute them evenly on the machines (essentially with some greedy algorithm) such that each machine gets at most $\frac{m\cdot\epsilon^2}{m\cdot\epsilon^{k+1}} = 1/\epsilon^{k-1}$ jobs with thus a total load of at most $\epsilon^k/\epsilon^{k-1} = \epsilon$. For the medium jobs of a small bag B we observe that up to $\epsilon \cdot m$ machines already have a large job from B assigned to them but we can still use at least $(1-\epsilon)m$ machines for the medium jobs from B. We obtain almost the same bound as above due to an assignment via a flow network that we use to round a fractional solution in which each bag distributes its medium jobs evenly among its at least $(1-\epsilon)m$ available machines.

▶ **Lemma 8.** *In polynomial time we can compute an assignment of the medium jobs of the small bags such that each machine gets at most $2/\epsilon^{k-1}$ medium jobs with a total load of at most $O(\epsilon)$ and no machine has two medium or a medium and a large job from the same bag.*

2.4 Assignment of small jobs

It remains to assign all small jobs, from the large as well as from the small bags. Note that at this point it is not even clear that after the assignment of the large and medium jobs we can add the small jobs such that the overall makespan is $1 + O(\epsilon)$ (recall the example in Figure 1). Therefore, we prove this in the following lemma.

▶ **Lemma 9.** *Given the previously computed assignment of large and medium jobs, there exists an assignment of the small jobs to the machines such that the overall makespan is bounded by $1 + O(\epsilon)$.*

Proof. Consider the (possibly infeasible) schedule S in which the small jobs are assigned as in the optimal solution and the large and medium jobs are assigned as in our so far computed solution. Let B be a bag. There might be a machine i such that two jobs of B are assigned to i in S. Note that B has to be a small bag. Let m' be the number of these machines and call these machines and the corresponding small jobs *problematic*. Assume w.l.o.g. that B contains exactly m jobs (if not then we can add some small dummy jobs of zero length). Then there must be m' machines on which no job of B was assigned, we call these machines *free*. We take the problematic small jobs of B from their (problematic) machines and distribute them on the free machines such that no two small jobs are assigned to the same machine. We do this operation for each bag. Denote by S' the resulting schedule. We argue that our operation did not increase the load on each machine by more than $O(\epsilon)$. Suppose that we moved a problematic job j in a small bag B from some machine i to some machine i'. Since i' did not have any job from B assigned to it in S and each bag has exactly m jobs, this means that in OPT machine i' must have a medium or a large job from B assigned to it. If i' has a large job j' from B in OPT then we charge p_j to $p_{j'}$, using that $p_j < \epsilon p_{j'}$. If i' has a medium job j'' from B in OPT then we charge p_j to $p_{j''}$. Recall that we assigned the medium jobs of the small bags to the machines such that the load of each machine due to medium jobs in small bags is $O(\epsilon)$ (see Lemma 8). Hence, we can still use $p_{j''}$ to pay for one other job assigned to i' in S'.

For a machine i let $OPT_i^{\text{large}}, OPT_i^{\text{med}}, OPT_i^{\text{small}}$ denote the load in OPT due to large, medium, and small jobs, respectively, and by S_i^{med} the load due to the medium jobs in S. Thus, in S' the load of machine i is bounded by

$$OPT_i^{\text{large}} + OPT_i^{\text{med}} + OPT_i^{\text{small}} + \epsilon OPT_i^{\text{large}} + S_i^{\text{med}} \leq (1 + O(\epsilon))OPT.$$

where OPT_i^{large} bounds the load due to large jobs, $OPT_i^{\text{med}} + \epsilon OPT_i^{\text{large}}$ bounds the load due to medium jobs in large bags and reassigned small jobs from small bags, OPT_i^{small} bounds the load from non-reassinged small jobs, and S_i^{med} bounds the load from medium jobs in small bags. ◀

In order to compute an assignment of the small jobs, observe that after assigning the large and medium jobs the machines have only $O_\epsilon(1)$ different loads since there are at most $1/\epsilon^{k+1}$ jobs on each machine that are large or medium and the size of each of them comes from a set of only $O(\log_{1+\epsilon} 1/\epsilon^{k+1})$ different values. Thus, we can partition the machines into $O_\epsilon(1)$ different groups such that two machines in the same group have the same load and their medium and large jobs from large bags come from exactly the same set of large bags (there are only $O_\epsilon(1)$ possibilities for the latter property). We devise a dynamic program that assigns the small jobs to these *groups* of machines, rather than directly to machines. This DP ensures that the average load of a machine in each group is at most $1 + O(\epsilon)$ and that for each machine group and each bag B we can schedule all small jobs in B assigned to this group on its machines without violating the bag-constraint. Formally, assume that the machines are divided into groups $M_1, ..., M_{K''}$ with the above properties where $K''_\epsilon = O_\epsilon(1)$ and let α_s denote the load due to medium and large jobs of each machine in group M_s. Let β_s denote the load of the small jobs assigned to machines in group M_s. For each bag B and each machine group M_s denote by $M_s^B \subseteq M_s$ the machines in M_s that do not have a medium or large job from B assigned to it.

▶ **Lemma 10.** *There is a polynomial time algorithm that assigns the small jobs to the machine groups $M_1, ..., M_{K''}$ such that for each $s \in [K'']$ we have that from each bag B at most $|M_s^B|$ jobs are assigned to M_s and $\alpha_s + \frac{\beta_s}{|M_s|} \leq 1 + O(\epsilon)$. The algorithm runs in time $b^2(\frac{mn}{\epsilon})^{O(K'')}$, where $K'' = (\frac{1}{\epsilon})^2 \log(\frac{1}{\epsilon})$.*

Once the jobs are assigned to the groups, we need to assign the jobs to the machines within each group. For the case without the bag-constraints, one can easily show that a simple greedy algorithm will ensure that the load of the small jobs will be almost equally distributed among the machines, i.e., the makespan of any two machines will differ by at most the size of one small job. In the setting of the bag-constraints this is no longer that easy. Consider a group M_s and let J_s denote the small jobs assigned to M_s due to Lemma 10. We group the small jobs by their respective bags, denote by $J_s^\ell := J_s \cap B_\ell$ for each bag B_ℓ. Assume w.l.o.g. that $|J_s^\ell| = |M_s^{B_\ell}|$ for each bag B_ℓ. In each iteration we assign all jobs from one bag as follows. Consider the ℓ-th iteration in which we assign the jobs in J_s^ℓ. We order the jobs in J_s^ℓ non-increasingly by length, i.e., assume that $J_s^\ell = \{j_1, j_2, ..., j_{|B_\ell|}\}$ such that $p_1 \geq p_2 \geq ... \geq p_{|B_\ell|}$. We order the machines in $M_s^{B_\ell}$ non-decreasingly by the total load that they obtained from jobs in $B_1, ..., B_{\ell-1}$ that we previously assigned to them. Let $i_1, ..., i_{|B_\ell|}$ be this order. Then for each $\ell' \in \{1, ..., |J_s^\ell|\}$ we assign job $j_{\ell'}$ to machine $i_{\ell'}$. We call this algorithm *bag-LPT*.

If for each bag B_ℓ we have that $M_s^{B_\ell} = M_s$ then we can again argue that at the end the load on any two machines differs by at most the size of one small job, i.e., ϵ^{k+1}. However, this is no longer the case if $M_s^{B_\ell} \neq M_s$ for some bag B_ℓ since then there is a machine $i \in M_s \setminus M_s^{B_\ell}$ that does not gspaet a small job from a bag B_ℓ. This happens if B_ℓ is a small bag and machine i already has a large or medium job from B_ℓ assigned to it. However, to each machine i we assigned in total at most $O(1/\epsilon^k)$ jobs in small bags that are medium or large: at most $O(1/\epsilon^k)$ large jobs since each large job has a size of at least ϵ^k and at most $O(1/\epsilon^{k-1})$ medium jobs due to Lemma 8. Hence there can be only $O(1/\epsilon^k)$ bags B_ℓ such that $i \in M_s \setminus M_s^{B_\ell}$. This allows us to bound the error due to the above by $O(1/\epsilon^k) \cdot \epsilon^{k+1} = O(\epsilon)$.

▶ **Lemma 11.** *For each group M_s bag-LPT assigns the small jobs such that each machine in M_s has a load of at most $\alpha_s + \frac{\beta_s}{|M_s|} + O(1/\epsilon^k) \cdot \epsilon^{k+1} \leq 1 + O(\epsilon)$.*

Hence, we assigned all large, medium, and small jobs such that each machine has a load of $1 + O(\epsilon)$. This completes the proof of Theorem 1.

3 Special Case of Restricted Assignment

In this section we present our 8-approximation algorithm for minimizing the makespan on unrelated machines under bag-constraints in the restricted assignment case where we additionally assume that all jobs in each bag B_ℓ can be assigned to the same set of machines.

Our starting point is the LP-relaxation for the minimization the makespan on unrelated machines as it was used in [16, 19] and we add additional inequalities for the bag-constraints to it. Let T be a guessed value of the optimal makespan. We define a linear program $LP(T)$ that models the problem of finding a solution with makespan T. Recall that for each job j there is a value p_j such that $p_{ij} \in \{p_j, \infty\}$ for each machine i. For each job j denote by M_j the set of machines i such that $p_{ij} = p_j$.

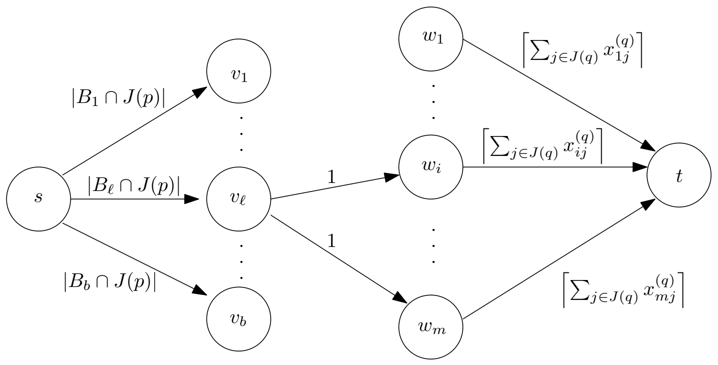
$$LP(T): \sum_{j: i \in M_j} x_{ij} p_{ij} \leq T \qquad \forall i \in M \tag{1}$$

$$\sum_{i \in M_j} x_{ij} = 1 \qquad \forall j \in J \tag{2}$$

$$\sum_{j \in B_\ell} x_{ij} \leq 1 \qquad \forall i \in M, \forall \ell \in [b] \tag{3}$$

$$x_{ij} \geq 0 \qquad \forall j \in J, i \in M_j \tag{4}$$

$$x_{ij} = 0 \qquad \text{if } p_{ij} > T, \forall j \in J, i \in M \tag{5}$$



$|B_1 \cap J(p)|$
$|B_\ell \cap J(p)|$
$|B_b \cap J(p)|$
s
v_1
v_ℓ
v_b
1
1
w_1
w_i
w_m
t
$\lceil \sum_{j \in J(q)} x_{1j}^{(q)} \rceil$
$\lceil \sum_{j \in J(q)} x_{ij}^{(q)} \rceil$
$\lceil \sum_{j \in J(q)} x_{mj}^{(q)} \rceil$

Figure 2 The flow-network for assigning the jobs in $J(p)$. Note that arcs of the type $\{v_\ell, w_i\}$ exist only if there is a job $j \in B_\ell \cap J(q)$ such that $x_{ij} > 0$. The values above the arcs indicate their respective capacities.

Using a binary search framework we determine $T^\star$ which we define to be the smallest value T for which $LP(T)$ is feasible. Denote by $x^\star$ the corresponding fractional solution. In the remainder of this section, we will prove that from $x^\star$ we can obtain an integral solution of makespan at most $4T^\star + 4\max_{i,j} p_{ij} \leq 8OPT$.

Assume w.l.o.g. that $p_{ij} \in \mathbb{N}$ for each machine $i \in M$ and each job $j \in J$. We round each finite job size p_{ij} and each value p_j to the next larger power of 2, denote by $\bar{p}_{ij}$ and $\bar{p}_j$ the new respective values. This increases the fractional load on each machine by at most a factor of 2. Based on this, we group the jobs into *classes*. A job j belongs to class q if $\bar{p}_j = 2^q$. We define $\mathrm{cl}(j)$ to be the class of a job j and $J(q)$ to be the set of all jobs of class q. Let $q_{\max}$ denote the highest class of a job in the instance. We compute our integral job assignment in phases, one phase for each job class in the order $q_{\max}, q_{\max} - 1, ..., 0$. In each phase q we determine an integral assignment of all jobs of class q.

Assume that we are given a fractional solution $\left\{x_{ij}^{(q)}\right\}_{i \in M, j \in J_{\leq}(q)}$ at the beginning of phase q that satisfies constraints (2)-(5) of $LP(T)$ for all jobs in $J_{\leq}(q)$ where for each q' we define $J_{\leq}(q') := \bigcup_{q'':q'' \leq q'} J(q'')$. In the first phase where $q = q_{\max}$ this solution $x^{(q)}$ equals the optimal LP-solution $x^\star$.

3.1 Job assignment via flow network

Similarly as in the proof of Lemma 8 we interpret the fractional assignment of the jobs in $J(q)$ given by $x^{(q)}$ as the fractional solution to an instance of maximum flow with integral edge capacities. Then, using flow theory we will argue that there exists also an integral solution to this instance which will then yield our integral job assignment.

Our (directed) flow-network consists of a source node s, a node v_ℓ for each bag B_ℓ, a node w_i for each machine i, and a sink node t (see Figure 2 for a sketch). For each bag B_ℓ there is an arc (s, v_ℓ) whose capacity equals the number of jobs in B_ℓ of class q, i.e., $|B_\ell \cap J(q)|$. For each bag B_ℓ and each machine i there is an arc (v_ℓ, w_i) of capacity 1 if and only if there is a job $j \in B_\ell \cap J(q)$ such that $x_{ij} > 0$. For each machine i there is an arc (w_i, t) whose capacity equals $\lceil \sum_{j \in J(q)} x_{ij}^{(q)} \rceil$, i.e., the fractional number of jobs of class q assigned to i, rounded up. Let $G^{(p)}$ be the resulting graph and denote by $(G^{(p)}, s, t)$ the overall flow network. The solution x yields a fractional flow in this network that sends $|J(q)|$ units of flow from s to t. Hence, standard flow theory implies the following proposition.

▶ **Proposition 12.** *There is an integral flow y for $(G^{(p)}, s, t)$ that sends $|J(q)|$ units of flow from s to t.*

The integral flow due to Proposition 12 yields an assignment of the jobs in $J(q)$ that respects the bag constraints: we assign a job from a bag B_ℓ to a machine i if and only if $y_{(v_\ell, w_i)} = 1$. This yields the following lemma.

▶ **Lemma 13.** *Given a class q and a solution $x^{(q)}$ that satisfies constraints (2)-(5) of $LP(T)$ for all jobs in $J_{\leq}(q)$. Then in polynomial time we can compute an integral assignment $\{\bar{x}_{ij}^{(q)}\}_{i\in M, j\in J(q)}$ for all jobs in $J(q)$ such that (i) each machine i has at most $\lceil \sum_{j\in J(q)} x_{ij}^{(q)} \rceil$ jobs of $J(q)$ assigned to it and (ii) if for some bag B_ℓ a job $j \in B_\ell \cap J(q)$ is assigned to a machine i then there is a job $j' \in B_\ell \cap J(q)$ with $x_{ij'}^{(q)} > 0$, and (iii) the solution $\bar{x}^{(q)}$ assigns at most one job from each bag to each machine.*

3.2 Reassignment of jobs

In the next phase $q-1$ we cannot directly apply the above procedure to assign the jobs in $J(q-1)$ starting with the solution $x^{(q)}$: it might be that there is a bag B_ℓ and two jobs $j, j' \in B_\ell$ such that $\mathrm{cl}(j) = q$, $\mathrm{cl}(j') = q-1$, j is assigned to some machine i in phase q, and $x_{ij'}^{(q)} > 0$. Hence, in phase $q-1$ potentially j' is also assigned to machine i which then violates the bag constraints. Therefore, based on $x^\star$ we construct a solution $\{x_{ij}^{(q-1)}\}_{i\in M, j\in J_{\leq}(q-1)}$ in which all jobs in $J_{\leq}(q-1)$ are fractionally assigned such that if $x_{ij}^{(q-1)} > 0$ for a job $j \in J_{\leq}(q-1)$ in some bag B_ℓ then there is no job $j' \in B_\ell$ with $\mathrm{cl}(j') \geq q$ that we assigned integrally to machine i in any of the previous phases. As we will see, this might increase the load on some machines, but only by a bounded amount.

Intuitively, suppose that $x_{ij}^\star > 0$ for some job $j \in J_{\leq}(q-1)$ in some bag B_ℓ and some machine i, and assume that in phase $q' \geq q$ we assigned a job $j' \in J(q') \cap B_\ell$ to machine i. We call such a pair (i, j) *problematic.* Then $x_{ij'}^\star \leq 1 - x_{ij}^\star$ due to constraint (3). Hence, at least a fraction of $x_{ij}^\star$ of job j' was assigned to some machine $i' \neq i$ by $x^\star$. Therefore, we can move $x_{ij}^\star$ units of job j to machine i' without violating the bag constraints. Even more, if job j is of class $\tilde{q}$ (note that $\tilde{q} \leq q-1$) then $\bar{p}_j \leq 2^{\tilde{q}-q'} \bar{p}_{j'}$, i.e., the additional load on machine i' is by a factor $2^{\tilde{q}-q'}$ smaller than the original load due to job j'.

More formally, initially we define $x^{(q-1)} := x^\star$. Consider a bag B_ℓ and denote by $\tilde{M}_\ell$ the set of all machines to which the jobs in B_ℓ can be assigned and to which we did not assign a job from B_ℓ in phases $q, ..., q_{\max}$. As long as there is a problematic pair (i, j) with $x_{ij}^{(q-1)} > 0$ we reassign the "problematic fraction" $x_{ij}^{(q-1)}$ of job j greedily to the machines in $\tilde{M}_\ell$ while we ensure that each machine $i \in \tilde{M}_\ell$ gets at most $\sum_{j\in B_\ell \cap (J \setminus J_{\leq}(q-1))} x_{ij}^\star$ jobs from B_ℓ reassigned overall (fractionally). Thus, at the end for each problematic pair (i, j) we have that $x_{ij}^{(q-1)} = 0$. We perform this reassignment for each bag B_ℓ.

▶ **Lemma 14.** *Given the integral assignment $\{\bar{x}_{ij}^{(q')}\}_{i\in M, j\in J(q')}$ of the jobs in $J(q')$ for each phase $q' \geq q$ due to Lemma 13. In polynomial time we can compute a (fractional) solution $\{x_{ij}^{(q-1)}\}_{i\in M, j\in J_{\leq}(q-1)}$ for the jobs in $J_{\leq}(q-1)$ such that*

(i) *$x^{(q-1)}$ satisfies constraints (2)-(5) of $LP(T)$,*

(ii) *for each job $j \in J(q-1)$ in some bag B_ℓ we have that $x_{ij}^{(q-1)} = 0$ if $\bar{x}_{ij'}^{(q')} = 1$ for some job $j \in J(q') \cap B_\ell$ for some $q' \geq q$,*

(iii) *for each machine i we have that*
$\sum_{j\in J(q-1)} x_{ij}^{(q-1)} \bar{p}_{ij} \leq \sum_{j\in J(q-1)} x_{ij}^\star \bar{p}_{ij} + \sum_{q' > q-1} 2^{(q-1)-q'} \sum_{j\in J(q')} x_{ij}^\star \bar{p}_{ij}.$

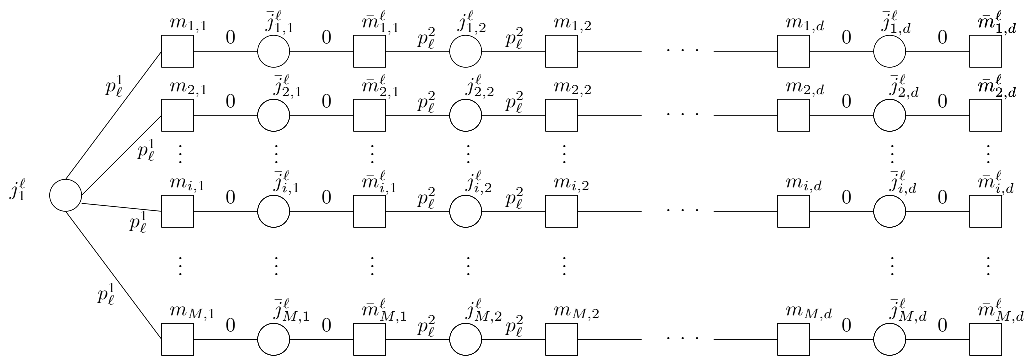


Figure 3 Sketch of the reduction from vector scheduling to group-restricted assignment.

We then proceed with phase $q-1$ where we start with the fractional solution $x^{(q-1)}$ as computed in Lemma 14. When we finish the last phase, we have computed an integral assignment $\{\bar{x}_{ij}\}_{i\in M, j\in J}$ of the jobs to the machines. Due to Lemma 13(ii) and Lemma 14(ii) our assignment respects the bag-constraints. In the next lemma we bound the load on each machine in the computed assignment which completes the proof of Theorem 3.

▶ **Lemma 15.** *In the computed assignment each machine has a load of at most* $4T^\star + 4\max_{i,j} p_{ij} \le 8T^\star \le 8OPT$.

4 Hardness of restricted assignment with bag-constraints

Our goal is to prove Theorem 2, i.e., we want to show that for minimizing the makespan on unrelated machines with bag-constraints there can be no $(\log n)^{1/4-\epsilon}$-approximation algorithm for any $\epsilon > 0$ unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(2^{(\log n)^{O(1)}})$, even for the restricted assignment case. We reduce the vector scheduling (VS) problem [3] to the problem of minimizing the makespan in the restricted assignment setting with bag-constraints. In the vector scheduling problem, we are given a set of identical machines M, a dimension $d \in \mathbb{N}$, and a set of n d-dimensional vectors $p_1, ..., p_n \in [0,\infty)^d$. The goal is to assign each vector to a machine, i.e., find a partition $A_1, ..., A_{|M|}$ of the vectors. The objective is to minimize $\max_{i\in M} \left\|\sum_{j\in A_i} p_j\right\|_\infty$. Our reduction is gap-preserving and in particular we will show that any c-approximation algorithm for our problem yields a c-approximation algorithm for vector scheduling for any value c. In [3] it was shown that the vector scheduling problem does not admit a c-approximation algorithm for any constant c, assuming that $\mathsf{P} \neq \mathsf{ZPP}$ (with the newer in approximability result for Independent Set in [23] it suffices to assume that $\mathsf{P} \neq \mathsf{NP}$). We are able to prove that one cannot get a $(\log n)^{1/4-\gamma}$-approximation for VS for any constant $\gamma > 0$ in polynomial time or quasi-polynomial time, assuming that $\mathsf{NP} \nsubseteq \mathsf{ZPTIME}(2^{(\log n)^{O(1)}})$. Then the same inapproximability bound holds for our problem as well.

Given an instance I of vector scheduling, defined by a number of dimensions d, a set of M identical machines, and n vectors $p_1, ..., p_n$ where for each vector p_i we denote by p_i^k its size in the k-th dimension. Denote by $OPT(I)$ its optimal objective value. We define an instance of makespan minimization on unrelated machines with bag-constraints. For each combination of a machine i and a dimension k in I we introduce a machine $m_{i,k}$, see Figure 3 for a sketch. For each vector p_ℓ we introduce a set of jobs that form a group J_ℓ. There is one job j_1^ℓ with size p_ℓ^1. The job j_1^ℓ can be assigned to each machine $m_{i,1}$ for each i. Intuitively, if j_1^ℓ is assigned to machine $m_{i,1}$ this corresponds to assigning the vector p_ℓ to machine i in I.

We will design the remaining machines and jobs for vector p_ℓ such that if j_1^ℓ is assigned to machine $m_{i,1}$ then for each dimension k

- each machine $m_{i,k}$ will get a load of p_ℓ^k from the jobs in J_ℓ and
- each machine $m_{i',k}$ with $i' \neq i$ will get a load of 0 from the jobs in J_ℓ.

Then, for each combination of a vector p_ℓ, a dimension $k \leq d$, and a machine i in I we introduce

- a dummy machine $\bar{m}_{i,k}^\ell$,
- a dummy job $\bar{j}_{i,k}^\ell$ of size 0 that can be assigned to only $m_{i,k}$ and $\bar{m}_{i,k}^\ell$, and
- if $k \geq 2$ then we also introduce a job $j_{i,k}^\ell$ of size p_ℓ^k that can be assigned to only $m_{i,k}$ and $\bar{m}_{i,k-1}^\ell$.

Observe that for each dummy machine $\bar{m}_{i,k}^\ell$ there are globally at most two jobs that can be assigned to it (and both are in J_ℓ). Denote by I' the resulting instance and denote by $OPT(I')$ its optimal solution value. Intuitively, we want to show that $OPT(I) = OPT(I')$ and that any solution $S(I)$ to I yields a solution $S(I')$ to I' with the same objective value. This needs some preparation. We have the following two lemmas.

▶ **Lemma 16.** *Let $\ell, i, k \in \mathbb{N}$. Consider any feasible solution. If the job j_1^ℓ is assigned to machine $m_{i,1}$ then each machine $m_{i,k}$ has exactly one job $j \in J_\ell$ assigned to it with $p_j = p_\ell^k$.*

The next lemma intuitively states that we can restrict ourselves to solutions to I' that are of the form described in the statement of the lemma.

▶ **Lemma 17.** *Consider any feasible solution S for the instance I' and let $i, \ell, k \in \mathbb{N}$. Further let job $j_{\ell,1}$ is assigned to machine $m_{i,1}$ in S. Then there exists another feasible solution S' such that each machine $m_{i',k}$ with $i' \neq i$ has at most one job from J_ℓ assigned to it and this job has size 0, while all other jobs have exactly the same assignment as that in S. Further, the makespan of S' is at most the makespan of S.*

▶ **Theorem 18.** *For any given instance I of the vector scheduling problem, in polynomial time we can construct an instance I' of the restricted assignment case of makespan minimization on unrelated machines with bag-constraints such that for any solution $S(I)$ for I there is a corresponding solution $S'(I')$ for I' with objective value at most that of $S(I)$ and vice versa. In particular, this implies that $OPT(I) = OPT(I')$.*

5 Conclusion

In this paper we showed that for minimizing the makespan on identical machines with bag-constraints there is a $(1+\epsilon)$-approximation algorithm like in the setting without the bag-constraints (and the problem is strongly NP-hard). However, we proved that for unrelated machines we see a change in complexity since in the classical setting the problem admits a $(2-1/m)$-approximation [19] while with the bag-constraints it seems unlikely to obtain a better approximation ratio than $(\log n)^{1/4-\epsilon}$ for any $\epsilon > 0$. It remains open to investigate more scheduling scenarios under bag-constraints such as makespan minimization on related machines or minimizing the weighted sum of completion time in any machine model. Also, for identical machines it remains open whether there is an EPTAS (which exists without the bag-constraints [12, 13, 10, 11, 17]).

Acknowledgments. We would like to thank the anonymous referees of this paper for many helpful comments and for pointing us to papers related to the setting of the bag-constraints.

References

1 Hans L. Bodlaender, Klaus Jansen, and Gerhard J. Woeginger. Scheduling with incompatible jobs. *Discrete Applied Mathematics*, 55(3):219–232, 1994.

2 Deeparnab Chakrabarty, Sanjeev Khanna, and Shi Li. On (1, ε)-restricted assignment makespan minimization. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1087–1101. SIAM, 2015.

3 Chandra Chekuri and Sanjeev Khanna. On multi-dimensional packing problems. In *SODA*, volume 99, pages 185–194. Citeseer, 1999.

4 Trivikram Dokka, Anastasia Kouvela, and Frits C. R. Spieksma. Approximating the multi-level bottleneck assignment problem. *Oper. Res. Lett.*, 40(4):282–286, 2012.

5 Tomáš Ebenlendr, Marek Krčál, and Jiří Sgall. Graph balancing: A special case of scheduling unrelated parallel machines. *Algorithmica*, 68(1):62–80, 2014.

6 Friedrich Eisenbrand, Karthikeyan Kesavan, Raju S. Mattikalli, Martin Niemeier, Arnold W. Nordsieck, Martin Skutella, José Verschae, and Andreas Wiese. *Solving an Avionics Real-Time Scheduling Problem by Advanced IP-Methods*, pages 11–22. Springer, 2010. `doi:10.1007/978-3-642-15775-2_2`.

7 Guy Even, Magnús M. Halldórsson, Lotem Kaplan, and Dana Ron. Scheduling with conflicts: online and offline algorithms. *J. Scheduling*, 12(2):199–224, 2009.

8 Ronald L Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

9 Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics*, 17(2):416–429, 1969.

10 D. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems.* PWS Publishing Company, 1997.

11 Dorit S Hochbaum and David B Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)*, 34(1):144–162, 1987.

12 Klaus Jansen. An eptas for scheduling jobs on uniform processors: using an milp relaxation with a constant number of integral variables. *SIAM Journal on Discrete Mathematics*, 24(2):457–485, 2010.

13 Klaus Jansen, Kim-Manuel Klein, and José Verschae. Closing the Gap for Makespan Scheduling via Sparsification Techniques. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *LIPIcs*, pages 72:1–72:13, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.ICALP.2016.72`.

14 Klaus Jansen, Kati Land, and Marten Maack. Estimating The Makespan of The Two-Valued Restricted Assignment Problem. In *15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2016)*, volume 53 of *LIPIcs*, pages 24:1–24:13, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.SWAT.2016.24`.

15 Klaus Jansen and Lars Rohwedder. On the configuration-lp of the restricted assignment problem. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*. SIAM, 2017. to appear.

16 Jan Karel Lenstra, David B. Shmoys, and Eva Tardos. Approximation algorithms for scheduling unrelated parallel machines. In *SFCS'87: Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 217–224, Washington, DC, USA, 1987. IEEE Computer Society. `doi:10.1109/SFCS.1987.8`.

17 Joseph YT Leung. Bin packing with restricted piece sizes. *Information Processing Letters*, 31(3):145–149, 1989.

18 Bill McCloskey and AJ Shankar. *Approaches to bin packing with clique-graph conflicts.* Computer Science Division, University of California, 2005.

19 E. V. Shchepin and N. Vakhania. An optimal rounding gives a better approximation for scheduling unrelated machines. *Operations Research Letters*, 33:127–133, 2005.

20 Mohit Singh. *Iterative methods in combinatorial optimization.* PhD thesis, Carnegie Mellon University, 2008.

21 Ola Svensson. Santa claus schedules jobs on unrelated machines. *SIAM Journal on Computing*, 41(5):1318–1341, 2012.

22 José Verschae and Andreas Wiese. On the configuration-lp for scheduling on unrelated machines. *Journal of Scheduling*, 17(4):371–383, 2014.

23 D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3:103–128, 2007.