ON THE CONFIGURATION-LP FOR SCHEDULING ON UNRELATED MACHINES

JOSÉ VERSCHAE AND ANDREAS WIESE

Abstract. One of the most important open problems in machine scheduling is the problem of scheduling a set of jobs on unrelated machines to minimize the makespan. The best known approximation algorithm for this problem guarantees an approximation factor of 2. It is known to be NP-hard to approximate with a better ratio than 3/2. Closing this gap has been open for over 20 years.

The best known approximation factors are achieved by LP-based algorithms. The strongest known linear program formulation for the problem is the *configuration-LP*. We show that the configuration-LP has an integrality gap of 2 even for the special case of *unrelated graph balancing*, where each job can be assigned to at most two machines. In particular, our result implies that a large family of cuts does not help to diminish the integrality gap of the canonical *assignment-LP*. Also, we present cases of the problem which can be approximated with a better factor than 2. They constitute valuable insights for constructing an NP-hardness reduction which improves the known lower bound.

Very recently Svensson [22] studied the *restricted assignment* case, where each job can only be assigned to a given set of machines on which it has the same processing time. He shows that in this setting the configuration-LP has an integrality gap of $33/17 \approx 1.94$. Hence, our result imply that the unrelated graph balancing case is significantly more complex than the restricted assignment case.

Then we turn to another objective function: maximizing the minimum machine load. For the case that every job can be assigned to at most two machines we give a purely combinatorial 2-approximation which is best possible, unless $P = NP$. This improves on the computationally costly LP-based $(2+\varepsilon)$-approximation algorithm by Chakrabarty et al. [7].

1. Introduction

The problem of minimizing makespan on unrelated machines, usually denoted $R||C_{\max}$, is one of the most prominent and important problems in the area of machine scheduling. In this setting we are given a set of n jobs and a set of m unrelated machines to process the jobs. Each job j requires $p_{i,j} \in \mathbb{N}^+ \cup \{\infty\}$ time units of processing if it is assigned to machine i. The scheduler must find an assignment of all jobs to machines with the objective of minimizing the makespan, i. e., the largest completion time of a job.

In a pioneering work, Lenstra, Shmoys, and Tardos [15] give a 2-approximation algorithm based on a natural LP-relaxation. On the other hand, they show that the problem is NP-hard to approximate within a factor better than 3/2, unless $P = NP$. Reducing this gap is considered one of the most important open questions in the area of machine scheduling [19] and it has been opened for more than 20 years.

Ebenlendr, Krcal, and Sgall [8] introduce a special case called the *graph balancing problem*. In this problem each job has finite processing time in only two given machines, and the processing times on both machines are the same. They give a 1.75-approximation algorithm based on an tighter version of the LP-relaxation by Lenstra et al. [15]. They strengthen this LP by adding inequalities that prohibit two large jobs to be simultaneously assigned to a machine.

In this paper we study several approaches to reduce the approximation gap. In particular we try to explain why it is so difficult to improve the 2-approximation factor for the general case. In the same line as in the work by Ebenlendr et al. [8], considerable effort has been given to add meaningful cuts to the LP-relaxation of Lenstra et al. [15], with the objective of reducing its integrality gap. We show that a large class of cuts, namely every family of cuts that involves only one machine per inequality, cannot improve the gap of 2. We show this by proving that the integrality gap of the so called *configuration-LP* is 2. Our construction works on the very restricted setting of the *unrelated graph balancing problem*, where each job can only be assigned to two different machines. Let us remark that, in contrast to the graph balancing setting, in the unrelated graph balancing problem the processing times of a job may be different in the two machines that a job has available.

In the second part of this paper we consider another related problem that has been in the eyes of the scheduling community in recent years. In the Min-Max allocation problem we are also given a set of jobs, a set of unrelated machines and processing times $p_{i,j}$ as before. The load of a machine i, denoted ℓ_i, is the sum of the processing times assigned to machine i. The objective is to maximize the minimum load of the machines, i.e., to maximize $\min_i \ell_i$. The idea behind this objective function is a fairness property: Consider that jobs represent resources that must be assigned to machines. Each machine i has a personal valuation of job (resource) j, namely $p_{i,j}$. The objective of maximizing the minimum machine load is equivalent to maximize the total valuation of the machine that receives the less resources.

1.1. **The minimum makespan problem.**

Unrelated machines. Besides the paper by Lenstra et al. [15] that we have already mentioned, there has not been much progress on how to diminish the approximation gap for $R||C_{\max}$. Shchepin and Vakhania [20] give a more sophisticated rounding for the LP by Lenstra et al. and improved the approximation guarantee to $2-1/m$, which is best possible among all rounding algorithms for this LP. On the other hand, Gairing, Monien, and Woclaw [10] propose a more efficient combinatorial 2-approximate algorithm based on unsplittable flows techniques.

Restricted assignment. A special case that has also been studied in the past is the *restricted assignment* problem. In this setting each job can only be assigned to a subset of machines, and has the same processing time in all its available machines. That is, the processing times of a job $p_{i,j}$ equals either a processing time independent of the machine $p_j \in \mathbb{N}^+$, or $p_{i,j} = \infty$. Although this problem seems to be much simpler that the unrelated machine problem, the best approximation factor known so far is $2-1/m$ that follows from the more general problem. In a very recent result Svensson [22] shows how to obtain in polynomial time and estimate of the optimal makespan that is within a factor $33/17 + \varepsilon$ to the optimal makespan. This is done by showing that the configuration-LP has an integrality gap of at most $33/17$. However no polynomial time rounding procedure is known.

People have also studied several special cases depending on the structure of the set of machines that the jobs can be assigned to, see [16] for a survey. Also, special cases concerning the processing times have been studied. In particular, Lin and Li [17] prove that if all processing times are equal the restricted assignment problem is solvable in polynomial time.

Graph balancing. The graph balancing problem can also be interpreted as a problem on an undirected graph. The nodes of the graph correspond to machines and edges correspond to jobs. The endpoints of an edge associated to job j are the machines on which j has finite processing time. The objective is to find an orientation of the edges so as to minimize the maximum load of all nodes, where the load of a node is defined as the sum of processing time of its incoming edges (jobs). Notice that the graph may have loops and in that case the corresponding job must be assigned to one particular machine.

Additionally to the 1.75-approximation algorithm for graph balancing presented by Ebenlendr et al. [8], they also show that it is NP-hard to approximate this problem within a factor better than $3/2$, matching the lower bound for the more general problem $R||C_{\max}$. On the other hand, some special cases have been studied. For example, it has been shown that if the underlying graph is a tree the problem admits a PTAS and if the processing times are either 1 or 2 the problem admits a $3/2$-approximation algorithm, which is tight. For these and more related results see [14] and the references therein.

1.2. The MaxMin-allocation problem.

Unrelated machines. The MaxMin-allocation problem has drawn a lot of attention recently. For the general setting of unrelated machines Bansal et al. [5] show that the configuration-LP has an integrality gap of $\Omega(\sqrt{m})$. On the other hand, Asadpour [3] show constructively that this is tight up to logarithmic factors and provide an algorithm with approximation ratio $O\left(\sqrt{m}\log^3 m\right)$. Relaxing the bound on the running time Chakrabarty et al. [7] present a poly-logarithmic approximation algorithm that runs in quasi-polynomial time. The best known NP-hardness result – even for the general case – shows that it is NP-hard to approximate the problem with a factor of $2-\epsilon$ for any $\epsilon > 0$ [6, 7]. For the special case that there are only two processing times arising in an instance (apart from zero), Golovin [11] gives a $O(\sqrt{n})$-approximation algorithm. He also provides an algorithm that gives at least a $(1-\frac{1}{k})$ fraction of the machines a load of at least OPT/k.

Restricted assignment. Bansal et al. [5] study the case that every job has the same processing time on every machine that it can be assigned to (restricted assignment case). They show that the configuration-LP has an integrality gap of $O(\log\log m/\log\log\log m)$ in this setting. Based on this they provide an algorithm with the same approximation ratio. The bound on the integrality gap was improved to $O(1)$ by Feige [9] and to 5 and subsequently to 4 by Asadpour et al. [2, 1]. The former proof is non-constructive using the Lovasz Local Lemma, the latter two are given by an (possibly exponential time) local search algorithm. However, Haeupler et al. [12] make the proof by Feige [9] constructive, yielding a polynomial time constant factor approximation algorithm.

Graph balancing. For the special case that every job can be assigned to at most two machines (but still with possibly different execution times on them) Bateni et al. [6] give a 4-approximation algorithm. Chakrabarty et al. [7] improve this by

showing that the configuration-LP has an integrality gap of 2, yielding a $(2+\epsilon)$-approximation algorithm. Moreover, it is NP-hard to approximate even this special case with a better ratio than 2 [6, 7]. Interestingly, the case that every job can be assigned to at most three machines is essentially equivalent to the general case [6].

1.3. **Our contribution.** As mentioned before, our main result for the minimum makespan problem is that the configuration-LP has an integrality gap of 2, even in the case of unrelated graph balancing. This implies that a set of cuts that involve only one machine per inequality cannot help to improve the integrality gap of the LP-relaxation of Lenstra, et al. [15]. Considering that the configuration-LP has a gap of $33/17 < 2$ [22], our result gives an indication that the real difficult instances of $R||C_{\max}$ corresponds to the the unrelated graph balancing and not the restricted assignment case. Additionally, we study special cases for which we obtain better approximation factors than 2. In particular we obtain a $1 + 5/6$ approximation guarantee for the special case of $R||C_{\max}$ when the processing times belong to the set $[\gamma, 3\gamma] \cup \{\infty\}$ for some $\gamma > 0$. Moreover, we show that there exists a $(2 - g/p_{\max} + \varepsilon)$-approximation algorithm, where g denotes the greatest common divisor of the processing times, and $p_{\max}$ the largest processing time. We also give a $5/3$-approximation algorithm for the case in which an optimal solution contains only a constant number of jobs that are larger than $2/3$ times the makespan. These results give necessary properties that an NP-hardness reduction must have to obtain an inapproximability factor 2.

We also consider restricted cases of the MaxMin-allocation problem. Our main result for this problem is in the unrelated graph balancing setting, for which we present a simple combinatorial algorithm that has a performance guarantee of 2 and quadratic running time. This improves on the LP-based $(2+\varepsilon)$-approximation algorithm by Chakrabarty et al. [7], which must use the ellipsoid method to approximately solve an LP with exponentially many variables and where the separation problem of the dual is a KNAPSACK problem and can only be solved approximately. Moreover, our algorithm exactly matches the lower bound of 2. Finally, we study what is achievable by allowing *half-integral* solutions, that is, solutions where we allow each job to be split in two halves. We give a polynomial time algorithm that computes solutions whose values are within a factor of 2 to the optimal integral solution. Moreover, by loosing an extra factor of 2 in the cost we can transform this solution to a solution with at most $m/2$ fractional jobs. This result contrasts the integral version of the problem for which only an $O(\sqrt{m}\log^3 m)$-approximation algorithm is known.

2. LP-Based approaches

In this section we revise the classical rounding procedure by Lenstra et al. [15] and elaborate on the implications of our results. In what follows we denote by J the set of jobs and M the set of machines of a given instance.

The natural LP relaxation. The natural IP-formulation used by Lenstra et al. [15] uses assignment variables $x_{i,j} \in \{0, 1\}$ that denote whether job j is assigned to machine i. This formulation, which we denote LST-IP, takes a target value for the makespan T (which will be determined later by binary search) and does not have

any objective function.

$$\begin{aligned} \sum_{i \in M} x_{i,j} &= 1 && \text{for all } j \in J, \\ \sum_{j \in J} p_{i,j} x_{i,j} &\leq T && \text{for all } i \in M, \\ x_{i,j} &= 0 && \text{for all } i,j : p_{i,j} > T, \\ x_{i,j} &\in \{0,1\} && \text{for all } i \in M, j \in J. \end{aligned}$$

It is not hard to see that this is indeed a formulation for $R||C_{\max}$. Indeed, the first equality ensures that all jobs must be completely assigned and the second guarantees that no machine has a load larger than T. The third equality is valid since no job can be assigned to a machine where its processing time is larger than T.

The corresponding LP-relaxation of this IP, which we denote LST-LP, can be obtained by replacing the integrality condition by $x_{i,j} \geq 0$. For a given value of T, this LP feasibility problem can be solved efficiently with the ellipsoid method.

Dual approximation. To obtain an approximation algorithm based on LST-LP, we need to guess the value of T by using a so called *dual approximation* technique, first introduced by Hochbaum and Shmoys [13]. Let C_{LP} be the smallest integer value of T so that LST-LP is feasible, and let C^* be the optimal value for the integral version of the LP (i.e., the optimal makespan of our instance). Note that since $p_{i,j} \in \mathbb{N}^+ \cup \{\infty\}$, then C_{LP} is a lower bound on C^*, and thus we can use it to design approximation algorithms. We can compute C_{LP} with the following procedure. If C_u is an upper bound and C_ℓ is a lower bound of C^*, by using a binary search procedure is easy to find C_{LP} by solving $\log_2(C_u - C_\ell)$ many times LST-LP. Indeed, we check at each iteration whether LST-LP is feasible for $T = (C_u + C_\ell)/2$. If this holds then we redefine $C_u := (C_u + C_\ell)/2$, and otherwise $C_\ell := (C_u + C_\ell)/2$. We iterate this until $C_u - C_\ell < 1$, and then notice that $C_{LP} = \lceil C_\ell \rceil$. By taking $C_u = \max_{i \in M} \sum_{j \in J} p_{i,j}$ and $C_\ell = 1$, we conclude that C_{LP} can be computed in polynomial time.

Rounding procedure. We now briefly present the 2-approximation of Lenstra et al. [15]. This procedure, which we call *LST-rounding*, takes a feasible solution of LST-LP with target makespan T and returns an integral solution with makespan at most $2T$. By taking $T = C_{LP} \leq C^*$ this yields a 2-approximation algorithm. The rounding procedure we present is a refinement of the original one, derived by Shmoys and Tardos [21] for the *generalized assignment* problem.

Let $(x_{i,j})$ be a fractional solution of LST-LP. We interpret such a solution as a fractional matching in an appropriate bipartite graph. For this we construct a bipartite graph $(A \cup B, E)$, where the vertices in A, also called the *job nodes*, corresponds to jobs. For each machine i, we construct $k_i = \left\lceil \sum_{j \in J} x_{i,j} \right\rceil$ nodes in set B, and call them $\{v_1^i, \ldots, v_{k_i}^i\}$. We refer to the nodes in B as the *machine nodes*.

We construct a fractional assignment of job nodes to machine nodes based on the assignment given by the $x_{i,j}$ variables. For this, fix a machine i and relabel the jobs so that $\{1, \ldots, n_i\}$ is the set of jobs that has some fraction assigned to i, i.e., $1 \leq j \leq n_i$ if and only if $x_{i,j} > 0$. Moreover, we assume w.l.o.g. that the jobs are in non-increasing order of processing times, $p_{i,1} \geq \ldots \geq p_{i,n_i}$. The fractional

assignment of jobs to the nodes of i is performed in a greedy fashion so that the larger jobs are assigned to the nodes of i of lower indexes. More precisely:

(1) Initialize $s = 0$.
(2) For each $j = 1, \ldots, n_i$:
   - If $\sum_{j'=j}^{n} y_{(v_s^i, j')} + x_{i,j} \leq 1$, then define $y_{(v_s^i, j)} := x_{i,j}$ and $y_{(v,j)} := 0$ for all other nodes v;
   - Otherwise

$$y_{(v_s^i, j)} := 1 - \sum_{j' \in J} y_{(v_s^i, j')},$$
$$y_{(v_{s+1}^i, j)} := x_{i,j} - y_{(v_s^i, j)},$$

$y_{(v_{s'}^i, j)} =: 0$ for all $s' \notin \{s, s+1\}$, and redefine $s := s + 1$.

Note that with this definition all job nodes are completely assigned to the machine nodes by the $y_{(v,j)}$ variables, i.e., $\sum_{v \in B} y_{(v,j)} = 1$ for all $j \in J$. Moreover, each node in B has at most one unit of job assigned to it, $\sum_{j \in J} y_{(v,j)} \leq 1$ for all $v \in B$. In other words, the variables $y_{(v,j)}$ defines a *fractional matching* that matches all job nodes. To obtain an integral assignment of jobs to machines, we simply define the set of edges E in our bipartite graph as the set $\{(j, v) \in A \times B : y_{(v,j)} > 0\}$. The rounding consists in finding a maximum matching in the constructed bipartite graph $(A \cup B, E)$, that matches each job node to some machine node. Such a matching exists since the existence of a fractional matching that matches all job nodes implies the existence of a matching satisfying the same property (see, e. g., [18, Vol A]). Each job j will be matched to some node v_s^i, and in this case we define the assignment variable $\bar{y}_{(v_s^i, j)} = 1$ and otherwise zero. Moreover, we define an assignment of jobs to machines as $\bar{x}_{ij} = \sum_{s=1}^{k_i} \bar{y}_{(v_s^i, j)}$ for all j and i.

Theorem 1 ([21])**.** *Let $(x_{i,j})_{j \in J, i \in M}$ be a feasible solution of LST-LP with target makespan T. Then, there exists a polynomial time rounding procedure that computes a binary solution $\{\bar{x}_{i,j}\}_{j \in J, i \in M}$ satisfying*

$$\sum_{i \in M} \bar{x}_{i,j} = 1 \qquad \text{for all } j \in J,$$
$$\sum_{j \in J} \bar{x}_{i,j} p_{i,j} \leq T + \max\{p_{i,j} : j \in J, x_{i,j} > 0\} \qquad \text{for all } i \in M.$$

Proof. We show that the previous rounding satisfies the properties of the lemma. Consider a machine i and a particular node v_s^i for $s \in \{2, \ldots, k_i\}$. In the rest of the proof we omit the super-index i to shorten notation. Note that the job assigned to node v_s by the matching $\bar{y}$ has processing time smaller or equal than all jobs that are fractionally assigned to v_{s-1}. Hence,

$$\sum_{j \in J} \bar{y}_{(v_s, j)} \cdot p_{i,j} \leq \sum_{j \in J} y_{(v_{s-1}, j)} \cdot p_{i,j}.$$

Summing over all $s \in \{2, \ldots, k_i\}$ we get that

$$\sum_{s=2}^{k_i} \sum_{j \in J} \bar{y}_{(v_s, j)} \cdot p_{i,j} \leq \sum_{s=2}^{k_i} \sum_{j \in J} y_{(v_{s-1}, j)} \cdot p_{i,j}.$$

Upper bounding $\sum_{j\in J} \bar{y}_{(v_1,j)} \cdot p_{i,j}$ by $\max\{p_{i,j} : j \in J, x_{i,j} > 0\}$ we conclude that

$$\begin{aligned}\sum_{j\in J} \bar{x}_{i,j}p_{i,j} &= \sum_{s=1}^{k_i}\sum_{j\in J} \bar{y}_{(v_s,j)} \cdot p_{i,j} \\ &\leq \sum_{s=1}^{k_i-1}\sum_{j\in J} y_{(v_s,j)} \cdot p_{i,j} + \max\{p_{i,j} : j \in J, x_{i,j} > 0\} \\ &\leq \sum_{j\in J} x_{i,j}p_{i,j} + \max\{p_{i,j} : j \in J, x_{i,j} > 0\}.\end{aligned}$$

□

By noting that $\max\{p_{i,j} : j \in J, x_{i,j} > 0\} \leq T$, the previous lemma yields that the rounding procedure embedded in a dual approximation framework is a 2-approximation algorithm for the makespan problem in unrelated machines.

Integrality gaps and the configuration-LP. Lenstra et al. [15] show that the rounding just given is best possible by means of the *integrality gap* of LST-LP. For an instance I of $R||C_{\max}$, let $C_{LP}(I)$ be the smallest integer value of T so that LST-LP is feasible, and let $C^*(I)$ the minimum makespan of this instance. Then the integrality gap of this LP is defined as

$$\max_I \frac{C^*(I)}{C_{LP}(I)}.$$

It is easy to see that if C_{LP} is used as a lower bound for deriving an approximation algorithm then the integrality is the best possible approximation guarantee that we can show. Lenstra et al. [15] give an example showing that the the integrality gap of LST-LP is as arbitrarily close to 2, and thus the rounding procedure is best possible.

It is natural to ask whether adding a family of cuts can help obtaining a formulation with smaller integrality gap. Indeed, it has been shown that for special cases of our problem adding inequalities reduces the integrality gap. In particular, Ebenlendr et al. [8] show that adding the following inequalities to LST-LP yields an integrality gap of at most 1.75 in the graph balancing setting,

$$\sum_{j\in J: p_{i,j} > T/2} x_{i,j} \leq 1 \qquad \text{for all } i \in M.$$

In this paper we study whether it is possible to add similar cuts to strengthen the LP for the unrelated graph balancing problem or for the more general case of $R||C_{\max}$. For this we consider the so called *configuration-LP*, defined as follows. Let T be a target makespan, and define $\mathcal{S}_i(T)$ as the collection of all subsets of jobs with total processing time at most T, i.e.,

$$\mathcal{C}_i(T) := \left\{ C \subseteq J : \sum_{j\in C} p_{i,j} \leq T \right\}.$$

We introduce a binary variable $y_{i,C}$ for all $i \in M$ and $C \in \mathcal{C}_i(T)$, representing whether the subset C of jobs is assigned to machine i. The LP is as follows:

$$\sum_{C \in \mathcal{C}_i(T)} y_{i,C} = 1 \qquad \text{for all } i \in M,$$

$$\sum_{i \in M} \sum_{C \in \mathcal{C}_i(T): C \ni j} y_{i,C} = 1 \qquad \text{for all } j \in J,$$

$$y_{i,C} \geq 0 \qquad \text{for all } i \in M, C \in \mathcal{C}_i(T).$$

It is not hard to see that an integral version of this LP is a formulation for $R||C_{\max}$. Also notice that the configuration-LP suffers from an exponential number of variables, and thus it is not possible to directly solve it in polynomial time. Moreover, using similar techniques as in [5], it is easy to show that the separation problem of the dual is equivalent to the KNAPSACK problem and thus we can solve the LP-approximately in polynomial time. More precisely, given a target makespan T for which the configuration-LP is feasible, it is possible to compute in polynomial time a solution of the configuration-LP with target makespan $T(1+\varepsilon)$, for any constant $\varepsilon > 0$ (see [22] for more details). The following result, which will be proven in the next section, shows that the integrality gap of this formulation is as large as the integrality gap of LST-IP even for the unrelated graph balancing case.

Theorem 2. *The integrality gap of the configuration-LP is 2 even in the unrelated graph balancing setting.*

Notice that a solution $(y_{i,C})$ of the configuration-LP can in interpreted in the $x_{i,j}$ variables of the LST-LP relaxation as follows,

$$x_{i,j} = \sum_{C \in \mathcal{C}_i(T): C \ni j} y_{i,C} \qquad \text{for all } i \in M, j \in J. \tag{2.1}$$

The converse is not true, since there are solutions to LST-LP that may not yield feasible solutions to the configuration-LP. For example, consider an instance with three jobs and two machines, where $p_{i,j} = 1$ for all jobs j and machines i. If we have a target makespan $T = 3/2$ is easy to see that LST-LP is feasible, but the solution space of the configuration-LP is empty for any $T < 2$. In what follows we make clear what is the relation of the two LPs, by giving a formulation in the space with $x_{i,j}$ variables that is equivalent to the configuration-LP.

Proposition 3. *Let $x^C \in \{0,1\}^J$ be the characteristic vector of a configuration $C \in \mathcal{C}_i(T)$, i.e., x^C_j is one if $j \in C$ and zero otherwise. There exists a feasible solution to the configuration-LP if and only if the following linear system admits a solution:*

$$\sum_{i \in M} x_{i,j} = 1 \qquad \textit{for all } j \in J, \tag{2.2}$$

$$(x_{i,j})_{j \in J} \in \operatorname{conv}\{x^C : C \in \mathcal{C}_i(T)\} \qquad \textit{for all } i \in M, \tag{2.3}$$

where $\operatorname{conv}\{S\}$ *denotes the convex closure of set* $S \in \mathbb{R}^n$.

Proof. Let $(x_{i,j})_{i \in M, j \in J}$ be a solution satisfying (2.2) and (2.3) for a given T. We show that the configuration-LP is feasible for the same value of T. Indeed, $(x_{i,j})_{j \in J}$

is a convex combination of vectors in $\{x^C : C \in \mathcal{C}_i(T)\}$, and thus

$$(x_{i,j})_{j \in J} = \sum_{C \in \mathcal{C}_i(T)} y_{i,C} \cdot x^C,$$

for some values $y_{i,C} \geq 0$ such that $\sum_{C \in \mathcal{C}_i(T)} y_{i,C} = 1$. Moreover, for each $j \in J$,

$$1 = \sum_{i \in M} x_{i,j} = \sum_{i \in M} \sum_{C \in \mathcal{C}_i(T)} y_{i,C} \cdot x_j^C = \sum_{i \in M} \sum_{C \in \mathcal{C}_i(T) : C \ni j} y_{i,C}.$$

This shows that $(y_{i,C})$ is a solution to the configuration-LP. The converse implication follows from reversing the argument just given. □

The last lemma implies that adding to LST-LP any family of cuts that does not remove any vector of the form $(x^{C_1}, \ldots, x^{C_m}) \in \mathbb{R}^{nm}$ where $C_i \in \mathcal{C}_i(T)$ cannot help reduce the integrality gap of the linear relaxation.

3. The Configuration-LP

We have seen in the previous section that the configuration-LP implicitly contains a vast class of linear cuts. Hence, it is at least as strong (in terms of its integrality gap) as any linear program that contains any subset of these cuts. However, in this section we prove that the configuration-LP has an integrality gap of 2. This implies that even all the cuts that are contained in the configuration-LP are not enough to construct an algorithm with a better approximation factor than 2.

Then we show that even for the special case of unrelated graph balancing the configuration-LP has an integrality gap of 2. This is somewhat surprising: if one additionally requires that each job has the same processing time on its two machines then Sgall et al. [8] implicitely proved that the configuration-LP has an integrality gap between 1.5 and 1.75. Hence, we demonstrate that the property that a job can have different processing times on different machines makes the problem significantly harder.

3.1. Integrality gap of the configuration-LP.

We describe a family of instances I_k for the general $R||C_{max}$-problem such that the configuration-LP has an integrality gap of $2 - \frac{1}{k}$ for each instance I_k. Since we can choose k arbitrarily large this proves an integrality gap of 2 of the configuration-LP. The construction we present here is significantly simpler than the construction for unrelated graph balancing which we will present in Section 3.2.

Let k be an integer. The instance I_k has $2k$ machines $m_1, m_1', m_2, m_2', ..., m_k, m_k'$. For every pair of machines m_i, m_i' there are k jobs $j_i^1, j_i^2, ..., j_i^k$ which have processing time $\frac{1}{k}$ on m_i, processing time 1 on m_i', and processing time ∞ on any other machine. There is one job j_{big} which has processing time 1 on any machine m_i and ∞ on any machine m_i'.

Lemma 4. *Every integral solution for I_k has a makespan of at least $2 - \frac{1}{k}$.*

Proof. Consider an integral solution for I_k and assume its makespan is less than 2. Let m_i be the machine that j_{big} is assigned to. At most one of the jobs $j_i^1, j_i^2, ..., j_i^k$ is assigned to m_i' and the other $k-1$ jobs are assigned to m_i. Hence, machine m_i has a makespan of $1 + (k-1) \cdot \frac{1}{k} = 2 - \frac{1}{k}$. □

Now let us study the configurations for the different machines. Since we want to show an integrality gap of $2-\frac{1}{k}$ we consider only configurations with makespan at most 1. Also, we consider only maximal configurations, i.e., configurations whose jobs are not all contained in another configuration on the respective machine. For each machine m_i there are two maximal configurations: all jobs j_i^ℓ or only j_{big}. We call the former configuration the *small configuration* and the latter the *big configuration*. For each machine m_i' there are k maximal configurations: each job j_i^ℓ for $1 \leq \ell \leq k$.

Lemma 5. *There is a solution of the configuration-LP for I_k that uses only configurations with makespan 1.*

Proof. Every machine m_i is assigned a ratio of $\frac{1}{k}$ of the big configuration and a ratio of $1-\frac{1}{k}$ of the small configuration. Note that this assigns the job j_{big} completely and every job j_i^ℓ is assigned to an extend of $1-\frac{1}{k}$. Every machine m_i' is assigned a ratio of $\frac{1}{k}$ of each of its k configurations. Hence, every job j_i^ℓ is now fully assigned. □

Theorem 6. *The configuration-LP for the $R||C_{\max}$-problem has an integrality gap of at least $2-\frac{1}{k}$ for instances such that $p_{i,j} \in \{\frac{1}{k}, 1, \infty\}$ for all machines i and all jobs j.*

The bound of $2-\frac{1}{k}$ is actually tight for instance where all $p_{i,j} \in \{\frac{1}{k}, 1, \infty\}$ as the following proposition shows.

Proposition 7. *The integrality gap of the configuration-LP for the $R||C_{\max}$-problem for instances with $p_{i,j} \in \{\frac{1}{k}, 1, \infty\}$ is bounded by $2-\frac{1}{k}$.*

Proof. Let I be an instances of $R||C_{\max}$ such that $p_{i,j} \in \{\frac{1}{k}, 1, \infty\}$ for all machines i and all jobs j. Assume that there is a feasible solution for the configuration-LP for I using only configurations with a makespan of at most T. If $T < 1$ then the LST-rounding procedure will yield a solution with makespan at most $T + \frac{1}{k}$. So now assume that $T = 1 + \ell\frac{1}{k}$ for some integer ℓ (other values for the makespan cannot arise). We perform the LST-rounding and analyze the resulting makespan. Consider a fixed machine i. We call a job j *big* if $p_{i,j} = 1$ and *small* if $p_{i,j} = \frac{1}{k}$. We call a configuration *big* if it contains a big job and *small* otherwise. We can assume that the configuration-LP assigned y_{big} units of big configurations to i and y_{small} units of small configurations. W.l.o.g. we assume that each assigned big configuration contains one big job and ℓ small jobs. In the rounding procedure $y_{big} \cdot (\ell+1) + y_{small} \cdot (\ell+k)$ vertices are introduced for i. After the rounding at most one of them can have a big job assigned to it. Hence, the total makespan of i is bounded by

$$\begin{aligned} 1 + \frac{1}{k}\left(y_{big} \cdot (\ell+1) + y_{small} \cdot (\ell+k) - 1\right) &= 1 + y_{big} \cdot \frac{\ell+1}{k} + y_{small} \cdot \frac{\ell+k}{k} - \frac{1}{k} \\ &\leq 1 + T - \frac{1}{k}. \end{aligned}$$

The integrality gap becomes maximal for $T = 1$. □

3.2. **Integrality gap for unrelated graph balancing.** Now we improve the result from the previous section and show that even for unrestricted graph balancing the integrality gap of the configuration-LP is 2. We construct a family of instances I_k such that $p_{i,j} \in \{\frac{1}{k}, 1, \infty\}$ for each machine i and each job j for some integer k.

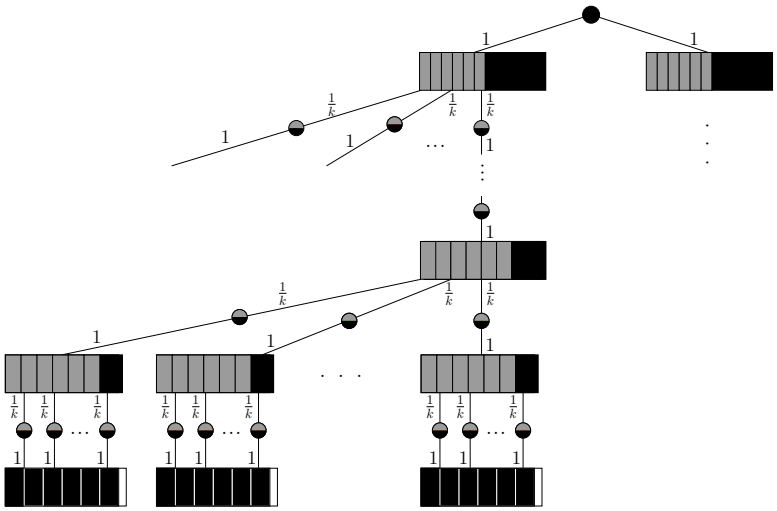

FIGURE 3.1. A sketch of the construction for the instance of unrelated graph balancing with an integrality gap of $2 - O(\frac{1}{k})$. The jobs on the machines correspond to the fractional solution of the configuration-LP for this instances with $T = 1 + \frac{1}{k}$.

We will show that for I_k there is a solution of the configuration-LP which uses only configurations with makespan $1 + \frac{1}{k}$. However, every integral solution for I_k has a makespan of at least $2 - \frac{1}{k}$.

Let $k \in \mathbb{N}$ and let N be the minimum integer such that $\left(\frac{k}{k-1}\right)^N \frac{1}{k-1} \geq \frac{1}{2}$. Consider two k-ary trees of height $N-1$, i.e., two trees of height $N-1$ in which apart from the leaves every vertex has k children. For every leaf v, we introduce another vertex v' and k edges between v and v'. (Hence, v is no longer a leaf.) Hence, the resulting "tree" has height N.

Based on this, we describe our instance of unrelated graph balancing. For each vertex v we introduce a machine m_v. For each edge $e = \{u, v\}$ we introduce a job j_e. Assume that u is closer to the root than v. We define that j_e has processing time $\frac{1}{k}$ on machine m_u, processing time 1 on machine m_v, and that it cannot be scheduled on any other machine. Finally, let $m_r^{(1)}$ and $m_r^{(2)}$ denote the two machines corresponding to the two root vertices. We introduce a job j_{big} which has processing time 1 on $m_r^{(1)}$ and $m_r^{(2)}$. Denote by I_k the resulting instance. See Figure 3.1 for a sketch.

As mentioned before, we claim that any integral solution for I_k has a makespan of at least $2 - \frac{1}{k}$. We prove this in the following lemma.

Lemma 8. *Any integral solution for I_k has a makespan of at least $2 - \frac{1}{k}$.*

Proof. Assume that we are given an integral solution for I_k which has a strictly smaller makespan than 2. W.l.o.g. assume that j_{big} is assigned to machine $m_r^{(1)}$. Since the makespan of our solution is strictly less than 2 at most $k-1$ jobs with processing time $\frac{1}{k}$ can be assigned to $m_r^{(1)}$. Hence, there is an edge e adjacent to the root r of the first tree such that j_e is *not* assigned to r. Thus, j_e must be assigned

to the machine corresponding to the other vertex that e is adjacent to. We iterate the argument. Eventually, we have that there must be a vertex v of height 1 and a corresponding machine m_v which has a job j with processing time 1 assigned to it. Recall that our solution has a makespan strictly less than 2. Hence, at most one job can be assigned to machine $m_{v'}$ where v' is the child vertex of v. Thus, $k-1$ jobs with processing time $\frac{1}{k}$ are assigned to m_v. Together with j this gives a makespan of $1+(k-1)\frac{1}{k} = 2-\frac{1}{k}$ on machine m_v. □

Now we want to show that there is a feasible solution for the configuration-LP for I_k which uses only configurations with makespan $1+\frac{1}{k}$. To this end, we introduce the concept of *j-α-solution* for the configuration-LP. We call j-α-solution a solution for the configuration-LP whose right hand side is modified as follows: the job j does not need to be fully assigned but only to an extend of $\alpha \leq 1$. I.e., instead of the equality

$$\sum_{i\in M}\sum_{C\in\mathcal{C}_i(T):C\ni j} y_{i,C} = 1$$

we have the inequality

$$\sum_{i\in M}\sum_{C\in\mathcal{C}_i(T):C\ni j} y_{i,C} \geq \alpha.$$

For any $h \in \mathbb{N}$ denote by $I_k^{(h)}$ a subinstance of I_k defined as follows: Take a vertex v of height h and consider the subtree $T(v)$ rooted at v. For the subinstance $I_k^{(h)}$ we take all machines and jobs which correspond to vertices and edges in $T(v)$. (Note that since our construction is symmetric it does not matter which vertex of height h we take.) Additionally, we take the job which has processing time 1 on m_v. We denote the latter by $j^{(h)}$.

We prove inductively that there are j-$\alpha^{(h)}$-solutions for the subinstances $I_k^{(h)}$ for values $\alpha^{(h)}$ which depend only on h. These values $\alpha^{(h)}$ increase for increasing h. The important point is that $\alpha^{(N)} \geq \frac{1}{2}$. Hence there are solutions for the configuration-LP which distribute j_{big} on the two machines $m_r^{(1)}$ and $m_r^{(2)}$ (which correspond to the two root vertices).

The following lemma gives the base case of the induction.

Lemma 9. *There is a $j^{(1)}$-$\frac{1}{k-1}$-solution for the configuration-LP for $I_k^{(1)}$ which uses only configurations with makespan at most $1+\frac{1}{k}$.*

Proof. Let m_v be the machine in $I_k^{(1)}$ which corresponds to the root of $I_k^{(1)}$. Similarly, let $m_{v'}$ denote the machine which corresponds to the leaf v'. For $\ell \in \{1,...,k\}$ let $j_\ell^{(0)}$ be the jobs which have processing time 1 on $m_{v'}$ and processing time $\frac{1}{k}$ on m_v.

For $m_{v'}$ the configurations with makespan at most $1+\frac{1}{k}$ are $C_\ell := \left\{j_\ell^{(0)}\right\}$ for each $\ell \in \{1,...,k\}$. We define $y_{m_{v'},C_\ell} := \frac{1}{k}$ for each ℓ. Hence, for each job $j_\ell^{(0)}$ a fraction of $\frac{k-1}{k}$ remains unassigned. For machine m_v there are the following (maximal) configurations: $C_{small} := \left\{j_1^{(0)},...,j_k^{(0)}\right\}$ and $C_{big}^\ell := \left\{j^{(1)}, j_\ell^{(0)}\right\}$ for each $\ell \in \{1,...,k\}$. We define $y_{m_v,C_{big}^\ell} := \frac{1}{k(k-1)}$ for each ℓ and $y_{m_v,C_{small}} :=$

$1 - \frac{1}{k-1}$. This assigns each job $j_\ell^{(0)}$ completely and the job $j^{(1)}$ to an extend of $k \cdot \frac{1}{k(k-1)} = \frac{1}{k-1}$. □

After having proven the base case, the following lemma yields the inductive step.

Lemma 10. *Assume that there is a $j^{(n)}$-$\left(\frac{1}{k-1}\left(\frac{k}{k-1}\right)^n\right)$-solution for the configuration-LP for $I_k^{(n)}$ which uses only configurations with makespan at most $1+\frac{1}{k}$. Then, there is a $j^{(n+1)}$-$\left(\frac{1}{k-1}\left(\frac{k}{k-1}\right)^{n+1}\right)$-solution for the configuration-LP for $I_k^{(n+1)}$ which uses only configurations with makespan at most $1+\frac{1}{k}$.*

Proof. Note that $I_k^{(n+1)}$ consists of k copies of $I_k^{(n)}$, one additional machine and one additional job. Denote by m_v the additional machine (which forms the "root" of $I_k^{(n+1)}$). Recall that $j^{(n+1)}$ is the (additional) job that can be assigned to m_v but to no other machine in $I_k^{(n+1)}$. For $\ell \in \{1, ..., k\}$ let $j_\ell^{(n)}$ be the jobs which have processing time $\frac{1}{k}$ on m_v.

Inside of the copies of $I_k^{(n)}$ we use the solution defined in the induction hypothesis. Hence, each job $j_\ell^{(n)}$ is already assigned to an extend of $\frac{1}{k-1}\left(\frac{k}{k-1}\right)^n$. Like in Lemma 9 the (maximal) configurations for m_v are $C_{small} := \left\{j_1^{(n)}, ..., j_k^{(n)}\right\}$ and $C_{big}^\ell := \left\{j^{(n+1)}, j_\ell^{(n)}\right\}$ for each $\ell \in \{1, ..., k\}$. We define $y_{m_v, C_{big}^\ell} := \frac{1}{k-1}\left(\frac{k}{k-1}\right)^n \frac{1}{k-1}$ for each ℓ and $y_{m_v, C_{small}} := 1 - \frac{k}{k-1}\left(\frac{k}{k-1}\right)^n \frac{1}{k-1}$. This assigns each job $j_\ell^{(n)}$ completely and the job $j^{(n+1)}$ to an extend of $k \cdot \frac{1}{k-1}\left(\frac{k}{k-1}\right)^n \frac{1}{k-1} = \frac{1}{k-1}\left(\frac{k}{k-1}\right)^{n+1}$. □

After this preparation we are ready to prove that there is a feasible solution for the configuration-LP for I_k.

Lemma 11. *There is a solution for the configuration-LP for I_k which uses only configurations with a makespan of at most $1 + \frac{1}{k}$.*

Proof. Recall that the two k-ary trees from the construction of I_k – together with the additional vertices – have height N such that $\left(\frac{k}{k-1}\right)^N \frac{1}{k-1} \geq \frac{1}{2}$. Hence, there are j_{big}-$\frac{1}{2}$-solutions for each of the two subinstances $I_k^{(N)}$ which use only configurations with a makespan of at most $1 + \frac{1}{k}$. This proves the claim. □

Now our main theorem follows from the previous lemmas.

Theorem 12. *The integrality gap of the configuration-LP for unrelated graph balancing is at least 2.*

Proof. Lemmas 8 and 11 imply that for the instance I_k the integrality gap of the configuration-LP is at least $(2 - \frac{1}{k})/(1 + \frac{1}{k})$. The claim follows since we can choose k arbitrarily large. □

4. Cases with Better Approximation Factors than 2

It has been open for a long time whether the approximation factor of 2 [15] for $R||C_{\max}$ can be improved. Our results from Section 3 can be seen as an indicator that this is not possible unless $P = NP$. In this section we identify classes of

instance for which a better approximation factor than 2 is possible. This can be understood as a guideline of properties that a NP-hardness reduction must fulfill to rule out a better approximation factor than 2.

4.1. **Instance with $p_{i,j} \in [\gamma, 3\gamma] \cup \{\infty\}$.** The NP-hardness proofs for $R||C_{\max}$ given in [8, 15] use only jobs such that $p_{i,j} \in \{1, 2, 3, \infty\}$. In this section we show that if the execution times of the jobs differ by at most a factor of 3 then the configuration-LP has an integrality gap of at most $1 + \frac{5}{6} \approx 1.83$. Recall that we can approximate the smallest value T for which the configuration-LP is feasible with an approximation factor of $1 + \epsilon$. Hence, there is a $(\frac{11}{6} + \epsilon)$-approximation algorithm for $R||C_{\max}$ in this setting. In particular, if one want to rule the existence of an $(2 - \epsilon)$-approximation algorithm one has to use instances with a larger discrepancy of the processing times.

We assume that there is a value γ such that $p_{i,j} \in [\gamma, 3\gamma] \cup \{\infty\}$ for all machines i and all jobs j.

Theorem 13. *Consider an instance of $R||C_{\max}$ with a value γ such that $p_{i,j} \in [\gamma, 3\gamma] \cup \{\infty\}$ for all machines i and all jobs j. Then for this instance the configuration-LP has an integrality gap of at most $1 + \frac{5}{6} \approx 1.83$.*

Proof. Assume we are given a value T such that there is a solution for the configuration-LP that uses only configurations with a makespan of at most T. We perform LST-rounding.

Consider a machine i. We want to bound the makespan of i. Note that if $T \geq \frac{18}{5}\gamma$ then the makespan of i is bounded by $\frac{18}{5}\gamma + 3\gamma \leq \left(1 + \frac{5}{6}\right) T$ since the makespan of each machine can increase by at most 3γ during the rounding process. So now assume that $T < \frac{18}{5}\gamma$. We define $x_{i,j}$ to be the fraction of job j that the configuration-LP assigned to i (see Equality 2.1).

Recall that the configuration-LP contains all linear cuts which are valid for all integral solutions. In particular, this implies that $\sum_{j:p_{i,j} > \frac{T}{2}} x_{i,j} \leq 1$ and $\sum_{j:p_{i,j} > \frac{T}{3}} x_{i,j} \leq 2$ (see Proposition 3).

Since $T < \frac{18}{5}\gamma < 4\gamma$ and $p_{i,j} \in [\gamma, 3\gamma] \cup \{\infty\}$ for all i, j we conclude that each configuration assigned to i can contain at most three jobs. Hence, in the LST-rounding procedure there are at most three vertices for i. Furthermore, for all jobs j connected to the ℓ-th vertex of i we have that $p_{i,j} \leq \frac{T}{\ell}$ for $\ell \in \{1, 2, 3\}$. This yields a bound $T + \frac{T}{2} + \frac{T}{3} = T(1 + \frac{5}{6})$ for the makespan of i. □

4.2. **Instances with bounded $\gcd_{i,j}\{p_{i,j}\}/\max_{i,j}\{p_{i,j}\}$.** In the known NP-hardness reductions for $R||C_{\max}$ [8, 15] only a fixed set of execution times arise (namely $\{\frac{1}{3}, \frac{1}{2}, 1\}$ or even only $\{\frac{1}{2}, 1\}$). In this section we give a result for classes of instances where only a finite set P of processing times arise. We prove that for such classes the configuration-LP has an integrality gap of at most $2 - \alpha$ for a value α which depends only on the processing times in P. Hence, if one wants to prove that $R||C_{\max}$ cannot be approximated with a better factor than 2, it is crucial to give a family of reduction with an infinite set of arising processing times.

W.l.o.g. we assume that all $p_{i,j}$ are always integers.

Theorem 14. *There exists a $(2 - \alpha)$-approximation algorithm for the problem of minimizing makespan on unrelated machines, where $\alpha = \gcd_{i,j:p_{i,j}<\infty}\{p_{i,j}\}/\max_{i,j:p_{i,j}<\infty}\{p_{i,j}\}$.*

Proof. We follow similar lines as the 2-approximation algorithm by Lenstra et al. [15]. Let $g := \gcd_{i,j:p_{i,j}<\infty}\{p_{i,j}\}$ and $M := \max_{i,j:p_{i,j}<\infty}\{p_{i,j}\}$. Note that the optimal makespan of our instance is a multiple of g, and therefore we can restrict our target makespan T to be of the form $k \cdot g$ with $k \in \mathbb{N}$. Let T^* be the target makespan defined as the smallest multiple of g that yields a feasible solution to LST-LP. Note that T^* can be found by a binary search procedure. Assume we have computed a fractional solution for LST-LP with target makespan T^*. We apply LST-rounding to this fractional solution, obtaining a schedule with load ℓ_i on each machine i. With basically the same argument as in the proof of Theorem 1, it is easy to see that $\ell_i < T^* + M$. Since ℓ_i, M and T^* are multiples of g, we conclude that $\ell_i \leq T^* + M - g$.

This proves that the makespan of machine i after LST-rounding is bounded by

$$\begin{aligned} T^* + M - g &\leq T^* \left(2 - \frac{(\beta+1)g}{T^*}\right) \\ &= T^* \left(2 - \frac{(\beta+1)g}{M + \beta \cdot g}\right) \\ &\leq T^* \left(2 - \frac{g}{M}\right) \\ &= T^* \left(2 - \alpha\right). \end{aligned}$$

□

In particular, the above theorem applies to families of instances which use only a finite set of processing times. Such families often arise in NP-hardness reductions. Hence, if one wants to prove that $R||C_{\max}$ cannot be approximated with a better factor than 2 then one has to construct reductions which use an infinite number of processing times. We formalize this observation in the following corollary.

Corollary 15. *Let $\mathcal{I}$ be a family of instances of $R||C_{\max}$. Let P be a finite set of integers. Assume that for each instance $I \in \mathcal{I}$ and each in I arising processing time $p_{i,j}$ it holds that $p_{i,j} \in P \cup \{\infty\}$. Then for $\mathcal{I}$ there is an approximation algorithm with performance guarantee $2 - \alpha$ with $\alpha = \gcd\{p|p \in P\}/\max\{p|p \in P\}$.*

4.3. **Bounded number of big jobs.** In this section we study the special case that in an optimal solution there are at most c big jobs, for some constant c. For this case we give a 5/3-approximation algorithm. Our method also yields a 5/3-approximation for the case that for at least $m - c$ machines it is known whether or not in an optimal solution they execute a big job which is big on that machine.

We call a job j *big on machine* i if $p_{i,j} \geq \frac{2}{3}OPT$. First, we present an algorithm that assumes that for each machine it is known in advance whether it executes a big job. Let $c \in \mathbb{N}$ be a fixed constant. Let I be an instance of $R||C_{\max}$. Assume that the set of machines is partitioned into two sets M_{big} and M_{small} such that we know that in some optimal solution each machine $i \in M_{big}$ executes a big job and each machine $i' \in M_{small}$ does not execute a big job. We use a binary search framework to "guess" the optimal makespan T. In the sequel, we describe one iteration of the binary search for a fixed value T.

For each machine $i \in M$ denote by $J^i_{big} \subseteq J$ all jobs j with $2T/3 \leq p_{i,j} \leq T$, by $J^i_{med} \subseteq J$ all jobs j with $T/3 \leq p_{i,j} < 2T/3$ and by $J^i_{small} \subseteq J$ all jobs j with $p_{i,j} < T/3$. We solve the following linear program LP_{bs}:

$$\begin{aligned}
\sum_{j \in J^i_{big} \cup J^i_{med}} x_{i,j} &\leq 1 && \forall i \in M_{big} \\
\sum_{j \in J^i_{small}} x_{i,j} \cdot p_{i,j} &\leq T/3 && \forall i \in M_{big} \\
\sum_{j \in J^i_{med} \cup J^i_{small}} x_{i,j} \cdot p_{i,j} &\leq T && \forall i \in M_{small} \\
\sum_{j \in J^i_{big}} x_{i,j} &= 0 && \forall i \in M_{small} \\
\sum_{i \in M} x_{i,j} &= 1 && \forall j \in J \\
x_{i,j} &\geq 0 && \forall i \in M, j \in J
\end{aligned}$$

Note that despite the separation of the jobs into the three classes for all $T \geq OPT$ the integral optimum satisfies LP_{bs} and hence LP_{bs} is feasible. Now assume that LP_{bs} is feasible for the guessed makespan T and let x be a solution. We perform LST-rounding.

Lemma 16. *Let I be an instance of $R||C_{\max}$, let T be an integer and assume we are given a partition of the machines into big and small machines. If LP_{bs} is feasible then the makespan after LST-rounding is bounded by $5T/3$.*

Proof. Let $i \in M_{small}$. Only jobs j with $p_{i,j} \leq 2T/3$ were (fractionally) assigned to i by LP_{bs}. Hence, during the LST-rounding the makespan of i can increase to at most $T + 2T/3 = 5T/3$. Now let $i' \in M_{big}$. The total processing time of small jobs on i' (bounded by $T/3$ in the LP-solution) can increase by at most $T/3$ (since $p_{i',j} \leq T/3$ for all $j \in J^{i'}_{small}$). There is at most one big job assigned to i'. Hence, the makespan of i' is bounded by $T/3+T/3+T = 5T/3$. Hence, the overall makespan of the solution is bounded by $5T/3$. □

For instances with at most c jobs which are big on some machine (for a fixed constant c) we can enumerate in polynomial time the at most $O(m^c)$ sets of machines which execute big jobs. For each of the sets we run the above algorithm together with a binary search framework. This yields the following theorem.

Theorem 17. *Let c be a fixed integer. There is a 5/3-approximation algorithm for instances of $R||C_{\max}$ with at most c jobs which are big on some machine.*

Moreover, for some instances it might be possible to determine for at least $m - O(\log n)$ machines whether they execute a big job or not (e.g., by some preprocessing). For such instances, our method also yields a 5/3-approximation.

Theorem 18. *Let c be a fixed integer. Let I be an instance of $R||C_{\max}$. Assume we know for at least $m - O(\log n)$ machines whether they execute big jobs. There is a polynomial time algorithm that computes a solution for I whose makespan is bounded by $\frac{5}{3}OPT$.*

Proof. For $m - O(\log n)$ machines we already know whether they are in M_{big} or M_{small}. For the remaining machines we enumerate whether they are big or small and use the above technique. □

The latter theorem is particularly important if one wants prove that there can be no approximation algorithm for $R||C_{\max}$ with a better factor than α for some $\alpha \in (\frac{5}{3}, 2]$: it shows that such a reduction must use instances for which no polynomial time algorithm can determine for almost all machines whether they execute a big job in an optimal solution.

5. MaxMin on Unrelated Machines

In this section we study the MaxMin-allocation problem on unrelated machines. First, we investigate the special case that every job can be assigned to at most two machines (MaxMin-balancing). For this case it is known that the configuration-LP has an integrality gap of 2. However, when allowing only polynomial running time it can only be solved approximately which yields a $(2+\epsilon)$-approximation algorithm for the overall problem. Also, it requires to solve a linear program with a PTAS for Knapsack as a separation oracle. In particular for small ϵ this algorithm requires a lot of running time and it is highly non-trivial to implement. Instead, we present here a purely combinatorial 2-approximation algorithm with a running time of $O(n^2)$ which is quite easy to implement.

After that we present approximation algorithms which compute 2- and 4-approximative half-integral solutions for the general MaxMin-allocation problem. Recall that for this setting the best known approximation algorithm (which computes integral solutions) has a performance guarantee of $O\left(\sqrt{m}\log^3 m\right)$.

5.1. **2-approximation for MaxMin-balancing.** We present our purely combinatorial 2-approximation algorithm for MaxMin-balancing. Let I be an instance of the problem and let T be a positive integer. Our algorithm either finds a solution with value $T/2$ or asserts that there is no solution with value T or larger. With an additional binary search this yields a 2-approximation algorithm. For each machine i denote by $J_i = \{j_{i,1}, j_{i,2}, ...\}$ the list of all jobs which can be assigned to i. We partition this set into the sets $A_i \dot\cup B_i$ where $A_i = \{a_{i,1}, a_{i,2}, ...\}$ denotes the jobs in J_i which can be assigned to two machines (machine i and some other machine) and B_i denotes the jobs in J_i which can only be assigned to i. We define A'_i to be the set A_i without the job with largest processing time. For any set of jobs J' we define $p(J') := \sum_{j\in J'} p_{i,j}$.

Denote by $p_{i,\ell}$ the processing time of job $a_{i,\ell}$ on machine i. We assume that the $a_{i,\ell}$ are ordered non-increasingly by processing time, i.e., $p_{i,\ell} \geq p_{i,\ell+1}$ for all respective values of ℓ. If there is a machine i such that $p(A_i)+p(B_i) < T$ we output that there is no solution with value T or larger. So now assume that $p(A_i)+p(B_i) \geq T$ for all machines i. If there is a machine i such that $p(A'_i) + p(B_i) < T$ (but $p(A_i) + p(B_i) \geq T$) then any solution with value at least T has to assign $a_{i,1}$ to i. Hence, we assign $a_{i,1}$ to i. This can be understood as moving $a_{i,1}$ from A_i to B_i. We rename the remaining jobs in A_i accordingly and update the values $p(A_i)$, $p(A'_i)$, and $p(B_i)$. We do this procedure until either

- there is one machine i such that $p(A_i) + p(B_i) < T$, in this case we output that there is no solution with value T or larger, or
- for all machines i we have that $p(A'_i) + p(B_i) \geq T$.

We call this phase the *preassignment phase*.

Lemma 19. *If during the preassignment phase the algorithm outputs that that no solution with value T or larger exists, then there can be no such solution.*

Proof. If the algorithm moves a job $a_{i,\ell}$ from A_i to B_i then any solution with value T or larger has to assign $a_{i,\ell}$ to B_i. Hence, if at some point there is a machine i such that $p(A_i) + p(B_i) < T$ then there can be no solution with value at least T. □

Now we construct a graph G as follows: For each machine i and each job $a_{i,\ell} \in A_i$ we introduce a vertex $\langle a_{i,\ell} \rangle$. We connect two vertices $\langle a_{i,\ell} \rangle, \langle a_{i',\ell'} \rangle$ if $a_{i,\ell}$ and $a_{i',\ell'}$ represent the same job (but on different machines). Also, for each machine i we introduce an edge between the vertices $\langle a_{i,2k+1} \rangle$ and $\langle a_{i,2k+2} \rangle$ for each respective value $k \geq 0$.

Lemma 20. *The graph G is bipartite.*

Proof. Since every vertex in G has degree two or less the graph splits into cycles and paths. It remains to show that all cycles have even length. There are two types of edges: edges which connect two vertices $\langle a_{i,\ell} \rangle, \langle a_{i',\ell'} \rangle$ such that $i = i'$ and edges connecting two vertices which correspond to the same job on two different machines. The edges of these two types alternate and hence the graph is bipartite. □

Due to Lemma 20 we can color G with two colors, black and white. Let i be a machine. We assign each the job $a_{i,\ell}$ to i if and only if $\langle a_{i,\ell} \rangle$ is black. Also, we assign each job in B_i to i.

Lemma 21. *The algorithm outputs a solution whose value is at least $T/2$.*

Proof. Let i be a machine. We show that the total weight of the jobs assigned to i is at least $p(A'_i)/2 + p(B_i)$. For each connected pair of vertices $\langle a_{i,2k+1} \rangle, \langle a_{i,2k+2} \rangle$ we have that either $a_{i,2k+1}$ or $a_{i,2k+2}$ is assigned to i. We calculate that $\sum_{k \in \mathbb{N}} p_{i,2k+2} \geq p(A'_i)/2$. Since $p_{i,2k+1} \geq p_{i,2k+2}$ (for all respective values k) we conclude that the total weight of the jobs assigned to i is at least $p(A'_i)/2 + p(B_i)$. Since $p(A'_i) + p(B_i) \geq T$ the claim follows. □

In order to turn the above algorithm into a 2-approximation algorithm a binary search is additionally necessary. Now we discuss how to implement the overall algorithm efficiently.

First, we test whether $n < m$. If this is the case then the optimal solution has value 0. So now assume that $n \geq m$. In order to initialize the ordered sets A_i and B_i we need to sort the jobs by execution time (in the list that we sort we have two entries for every job, each corresponding to one of its possible execution times). We sort this list in $O(n \log n)$ steps. Note that the sorting needs to be done only once, no matter how many values T we try. Starting with an ordered list of the jobs, we can build the ordered lists A_i and the sets B_i in linear time. The preassignment phase can be implemented in linear time: For each machine i we need to check whether $p(A'_i) + p(B_i) < T$. We call this a *first-check*. If we move a job $a_{i,\ell}$ from A_i to B_i then the other machine on which one could possibly assign $a_{i,\ell}$ needs to be checked again. We call this a *second-check*. There are m first-checks and at most n second-checks necessary. Hence, this procedure can be implemented in linear time. Coloring the graph G with two colors can also be implemented in linear time.

For the binary search we need to try at most $\log D$ values, with $D := \sum_{i,j} p_{i,j}$. We have that $\log D \leq |I|$ where $|I|$ denotes the length of the overall input in binary encoding. The sorting needs to be done only once and this takes in $O(|I| \log |I|)$

time. For every value T that we try $O(|I|)$ steps are necessary. This yields an overall running time of $O\left(|I|^2\right)$.

Theorem 22. *There is a 2-approximation algorithm for the Max-Min-balancing problem with running time $O\left(|I|^2\right)$.*

5.2. **Half-integral solutions.** For the general MaxMin-allocation problem the best known polynomial time approximation algorithm achieves an approximation factor of $O(\sqrt{m}\log^3 m)$ [4]. A constant factor approximation algorithm seems difficult to achieve, in particular since the configuration-LP has an integrality gap of $\Omega(\sqrt{m})$ [5]. However, we present a polynomial time algorithm that computes a half-integral solution whose value is at most by a factor 2 smaller than the best integral solution. Moreover, we show that at the cost of at most a factor of 2 this solution can be transformed to a half-integral solution in which at most $m/2$ jobs are fractionally assigned.

Let I be an instance of the MaxMin allocation problem. Let T be a guessed optimal value (later determined by a binary search). As a first step we redefine I to an instance I' by changing the execution times of the jobs to $p'_{i,j} := \min\{p_{i,j}, T\}$. Clearly, if there is a solution with value T for I then there is also a solution with value T for I'. With the instance I' we solve the following linear program LP_{as}:

$$\sum_j x_{i,j} \cdot p'_{i,j} \geq T$$

$$\sum_i x_{i,j} = 1$$

$$x_{i,j} \geq 0$$

If LP_{as} is infeasible there can be no integral solution with value T or larger. Now assume that LP_{as} is feasible. We perform a slightly modified LST-rounding: instead of assigning one unit of jobs to each vertex for a machine i, we assign $1/2$ units of jobs to every vertex (and hence we need more vertices). After the rounding we obtain a half-integral solution $HALF(I)$.

Theorem 23. *Let I be an instance of the MaxMin-allocation problem. There is a polynomial time algorithm that computes a half-integral solution $HALF(I)$ such that $HALF(I) \geq \frac{1}{2}OPT(I)$.*

Proof. Let i be a machine. Similarly to the analysis of LST-rounding for $R||C_{\max}$, during the rounding we lose at most the volume of the jobs assigned to the first vertex of i. Since at most $1/2$ units of jobs can be assigned to this vertex and all $p'_{i,j} \leq T$, machine i keeps a makespan of at least $T/2$. Since LP_{as} was feasible for T we conclude that T is an upper bound for $OPT_{half}(I)$ and $OPT(I)$. □

Now we show how to modify $HALF(I)$ to get another half-integral solution in which at most $m/2$ jobs are fractionally assigned. This modification comes at a cost of at most a factor 2 and hence still yields a constant factor approximation.

Let I be an instance of the MaxMin-allocation problem and let $HALF(I)$ be any half-integral solution for I. We do not change the assignment of jobs which were assigned integrally in $HALF(I)$. For each machine i let $J_i = \{j_{i,1}, j_{i,2}, ...\}$ denote the jobs which are fractionally assigned to i. Let $p_{i,\ell}$ denote the processing time of $j_{i,\ell}$. We assume that the jobs are ordered non-increasingly by processing time,

i.e., $p_{i,\ell} \geq p_{i,\ell+1}$ for all respective values of ℓ. We define a graph G as follows: For each job $j_{i,\ell}$ we introduce a vertex $\langle j_{i,\ell} \rangle$. We connect two vertices $\langle j_{i,\ell} \rangle$, $\langle j_{i',\ell'} \rangle$ by an edge if $j_{i,\ell}$ and $j_{i',\ell'}$ represent the same job (but on different machines). We call such edges the *outer edges*. Also, for each machine i we introduce an edge between the vertices $\langle j_{i,2k} \rangle$ and $\langle j_{i,2k+1} \rangle$ for each respective value $k \geq 1$ (*inner edges*).

Lemma 24. *The graph G is bipartite.*

Proof. Can be proven similarly as Lemma 20. □

Now we color G (greedily) with two colors, black and white. From the coloring we compute a new solution as follows: we take each the black vertex $\langle j_{i,\ell} \rangle$ and assign the job $j_{i,\ell}$ completely to machine i. Note that this is well-defined since if $\langle j_{i,\ell} \rangle$ then the other vertex that represents the job is white.

Now there are two cases that we treat separately. For each machine i we call the vertex $\langle j_{i,1} \rangle$ the *head vertex*. Let P be a path in G. If one of the end vertices of P is a head vertex and the other one is not a head vertex, then we (re-)color P such that the head vertex is black and do the job assignment as above. If both end vertices of P are head vertices then we take the head vertex $\langle j_{i,1} \rangle$ that was colored in white and assign one half of the job $j_{i,1}$ to machine i and the other half to the respective other machine. Denote by $HALF'(I)$ the resulting solution.

Lemma 25. *The solution $HALF'(I)$ is half-integral and at most $m/2$ jobs are fractionally assigned. Moreover, $HALF'(I) \geq \frac{1}{2} \cdot HALF(I)$.*

Proof. From the definition of the algorithm we conclude that for each machine i the job $j_{i,1}$ corresponding to the head vertex $\langle j_{i,1} \rangle$ of i is at least half assigned to i. For each pair of jobs $j_{i,2k}, j_{i,2k+1}$ (for $k \geq 1$) one of them is at least half assigned to i. Denote by $B(i)$ the jobs which were integrally assigned to i in $HALF(I)$. In $HALF'(I)$ the machine i has a makespan of at least $\sum_{j \in B(i)} p_{i,j} + \frac{1}{2} \sum_{k \geq 0} p_{i,2k+1}$. Since $p_{i,2k} \geq p_{i,2k+1}$ for all machines i an all values k we conclude that

$$\begin{aligned}
\sum_{j \in B(i)} p_{i,j} + \frac{1}{2} \sum_{k \geq 0} p_{i,2k+1} &\geq \sum_{j \in B(i)} p_{i,j} + \frac{1}{4} \sum_{k \geq 1} p_{i,k} \\
&\geq \frac{1}{2} \left(\sum_{j \in B(i)} p_{i,j} + \frac{1}{2} \sum_{k \geq 1} p_{i,k} \right) \\
&= \frac{1}{2} HALF_i(I)
\end{aligned}$$

where $HALF_i(I)$ denotes the makespan of i in $HALF(I)$.

The only case in which a job is fractionally assigned is when both end vertices of a path P are head vertices. Since every machine has only one head vertex, there can be at most $m/2$ such paths. Hence, the number of fractionally assigned jobs is bounded by $m/2$. □

Theorem 26. *Let I be an instance of the MaxMin-allocation problem. There is a polynomial time algorithm that computes a half-integral solution $HALF'(I)$ such that $HALF(I) \geq \frac{1}{4} OPT(I)$. Moreover, $HALF'(I)$ assigns at most $m/2$ jobs fractionally.*

We note that with similar techniques as above we can compute half-integral solutions for $R||C_{\max}$ which are by at most a factor of 3/2 larger than the best integral solutions.

5.3. **Tractable cases.** Similarly as for $R||C_{\max}$ if the number of big jobs is bounded by a constant c or if we knew what machines execute a big job in an optimal solution we can guarantee better approximation factors. Here, we define a job j to be *big on machine* i if $p_{i,j} \geq \frac{1}{2}OPT$. With similar techniques as in Section 4.3 we can prove the following theorems.

Theorem 27. *Let c be a fixed integer. There is a 2-approximation algorithm for instances of the MaxMin-allocation problem with at most c big jobs.*

Theorem 28. *Let c be a fixed integer. Let I be an instance of the MaxMin-allocation problem. Assume we know for at least $m - c$ machines whether they execute a big job in an optimal solution. There is a polynomial time algorithm that computes a solution for I whose value is at least $\frac{1}{2}OPT$.*

References

[1] A. Asadpour, U. Feige, and A. Saberi. Santa claus meets hypergraph matchings. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques (APPROX and RANDOM 2008)*, volume 5171 of *Lecture Notes in Computer Science*, pages 10–20. Springer, 2008.

[2] A. Asadpour, U. Feige, and A. Saberi. Santa claus meets hypergraph matchings. Technical report, Standford University, 2009. Available for download at `http://www.stanford.edu/~asadpour/publication.htm`.

[3] A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *Proceedings of the 39th annual ACM symposium on Theory of computing (STOC 2007)*, pages 114–121, 2007.

[4] A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *Proceedings of the 39th annual ACM symposium on Theory of computing (STOC 2007)*, pages 114–121, 2007.

[5] N. Bansal and M. Sviridenko. The santa claus problem. In *Proceedings of the 38th annual ACM symposium on Theory of computing (STOC 2006)*, pages 31–40, 2006.

[6] M. Bateni, M. Charikar, and V. Guruswami. Maxmin allocation via degree lower-bounded arborescences. In *Proceedings of the 41st annual ACM symposium on Theory of computing (STOC 2009)*, pages 543–552, 2009.

[7] D. Chakrabarty, J. Chuzhoy, and S. Khanna. On allocating goods to maximize fairness. In *Proceedings of the 50th Annual Symposium on Foundations of Computer Science (FOCS 2009)*, pages 107–116, 2009.

[8] T. Ebenlendr, M. Krčál, and J. Sgall. Graph balancing: a special case of scheduling unrelated parallel machines. In *Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms (SODA 2008)*, pages 483–490, 2008.

[9] U. Feige. On allocations that maximize fairness. In *Proceedings of the 19th Fannual ACM-SIAM symposium on Discrete algorithms (SODA 2008)*, pages 287–293, 2008.

[10] M. Gairing, B. Monien, and A. Woclaw. A faster combinatorial approximation algorithm for scheduling unrelated parallel machines. *Theoretical Computer Science*, 380:87–99, 2007.

[11] D. Golovin. Max-min fair allocation of indivisible goods. Technical report, Carnegie Mellon University, June 2005.

[12] B. Haeupler, B. Saha, and A. Srinivasan. New Constructive Aspects of the Lovasz Local Lemma. *ArXiv e-prints*, January 2010.

[13] Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.

[14] K. Lee, J. Y.-T. Leung, and M. L. Pinedo. A note on graph balancing problems with restrictions. *Information Processing Letters*, 110:24–29, 2009.

[15] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.
[16] J. Y.-T. Leung and C.-L. Li. Scheduling with processing set restrictions: A survey. *International Journal of Production Economics*, 116:251–262, 2008.
[17] Y. Lin and W. Li. Parallel machine scheduling of machine-dependent jobs with unit-length. *European Journal of Operational Research*, 156:261–266, 2004.
[18] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, Berlin, 2003.
[19] P. Schuurman and G. J. Woeginger. Polynomial time approximation algorithms for machine scheduling: Ten open problems. *Journal of Scheduling*, 2:203–213, 1999.
[20] E. V. Shchepin and N. Vakhania. An optimal rounding gives a better approximation for scheduling unrelated machines. *Operations Research Letters*, 33:127–133, 2005.
[21] D. B. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461–474, 1993.
[22] O. Svensson. Santa Claus Schedules Jobs on Unrelated Machines. *ArXiv e-prints*, November 2010.