J Sched (2014) 17:371–383
DOI 10.1007/s10951-013-0359-4

On the configuration-LP for scheduling on unrelated machines

José Verschae · Andreas Wiese

Received: 11 May 2012 / Accepted: 28 October 2013 / Published online: 10 November 2013

Abstract Closing the approximability gap between 3/2 and 2 for the minimum makespan problem on unrelated machines is one of the most important open questions in scheduling. Almost all known approximation algorithms for the problem are based on linear programs (LPs). In this paper, we identify a surprisingly simple class of instances which constitute the core difficulty for LPs: the so far hardly studied *unrelated graph balancing* case in which each job can be assigned to at most two machines. We prove that already for this basic setting the strongest LP-relaxation studied so far—the *configuration-LP*—has an integrality gap of 2, matching the best known approximation factor for the general case. This points toward an interesting direction of future research. For the objective of maximizing the minimum machine load in the unrelated graph balancing setting, we present an elegant purely combinatorial 2-approximation algorithm with only quadratic running time. Our algorithm uses a novel preprocessing routine that estimates the optimal value as good as the configuration-LP. This improves on the computationally costly LP-based algorithm by Chakrabarty et al. (Proceedings of the 50th Annual Symposium on Foundations of Computer Science (FOCS 2009), pp 107–116, 2009) that achieves the same approximation guarantee.

Keywords Machine scheduling · Integrality gap · Configuration-LP · Approximation algorithms

J. Verschae
Departmento de Ingeniería Industrial, University of Chile,
República 701, Santiago, Chile
e-mail: jverscha@ing.uchile.cl

J. Verschae
Center for Mathematical Modeling, University of Chile,
Blanco Encalada 2120, Santiago, Chile

A. Wiese (✉)
Max-Planck-Institut für Informatik, Campus E1 4,
66123 Saarbrücken, Germany
e-mail: awiese@mpi-inf.mpg.de

1 Introduction

The problem of minimizing the 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 jobs to machines with the objective of minimizing the makespan, i.e., the largest completion time of a job.

In a seminal work, Lenstra et al. (1990) 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 better factor than 3/2, unless $P = NP$. Reducing this gap is considered to be one of the most important open questions in the area of machine scheduling (Schuurman and Woeginger 1999) and it has been open for more than 20 years.

The best known approximation algorithm for this problem and its special cases are derived by linear programming techniques (Ebenlendr et al. 2008; Lenstra et al. 1990; Svensson 2012). A special role plays the *configuration-LP* (which has been successfully used for Bin-Packing (Karmarkar and Karp 1982) and other scheduling problems (Bansal and Sviridenko 2006; Svensson 2012). It is the strongest linear program for the problem considered in the literature and it implicitly contains a vast class of inequalities. In fact, for the most relevant cases of $R||C_{\max}$ (i.e., the general case, the restricted assignment case, and the graph balancing case), the best known approximation/estimation factors match the best known upper bounds on the integrality gap of the configuration-LP.

Given the apparent difficulty of this problem, researchers have turned to consider simpler cases. One special case that has drawn a lot of attention is the *restricted assignment* problem. In this setting, each job can only be assigned to a subset of machines, on which it has the same processing time. That is, the processing times $p_{i,j}$ of a job j equal either a machine-independent processing time $p_j \in \mathbb{N}^+$ or infinity. Surprisingly, the best known approximation algorithm for this problem continues to be the 2-approximation algorithm by Lenstra et al. (1990). Svensson (2012) shows that the configuration-LP has an integrality gap of at most $33/17 \approx 1.94$. Thus, it is possible to compute in polynomial time a lower bound that is within a factor $33/17 + \epsilon$ to the optimum. However, no polynomial time algorithm is known to construct an α-approximate solution for $\alpha < 2$.

The restricted assignment case seems to capture the complexity of the general case to a major extend. However, we show that worst case instances for the configuration-LP lie in the *unrelated graph balancing case*, where each job can be assigned to at most two machines, but with possibly different processing times on each of them. Together with Svensson's (2012) result, this indicates that the core obstacles for the state-of-the-art algorithmic methods for the general case already lie in the unrelated graph balancing case which motivates more research in this direction.

In the second part of this paper, we study a different objective function which has been actively studied by the scheduling community in recent years, see e.g., Asadpour et al. (2008), Asadpour and Saberi (2010), Bansal and Sviridenko (2006), and Chakrabarty et al. (2009). In the *MaxMin-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 by ℓ_i, is the total processing time of jobs assigned to 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 maximizing the total valuation of the machine that receives the least total valuation.

An extended abstract of this paper appeared in the proceedings of the European Symposium on Algorithms (ESA) 2010 (Verschae and Wiese 2011).

1.1 The minimum makespan problem

The problem of minimizing the makespan on unrelated machines is considered to be an important problem in machine scheduling. In the sequel, we discuss the literature for the general problem and the already mentioned special cases.

1.1.1 General setting

As mentioned above, in a seminal paper Lenstra et al. (1990) present a 2-approximation algorithm and prove that the problem is NP-hard to approximate within a factor of $3/2 - \epsilon$ for all $\epsilon > 0$. Besides this paper, there has not been much progress on improving the approximation ratio for $R||C_{\max}$. Shchepin and Vakhania (2005) give a more sophisticated rounding for the linear program by Lenstra et al. and improve the approximation guarantee to $2 - 1/m$, which is best possible among all rounding algorithms for this LP. On the other hand, Gairing et al. (2007) propose a more efficient combinatorial 2-approximation algorithm based on unsplittable flow techniques. If the number of machines is constant, Horowitz and Sahni (April, 1976) give a $(1 + \epsilon)$-approximation algorithm. Note that also in this setting the problem is NP-hard (follows from a straightforward reduction from PARTITION).

In the preemptive version of this problem, we are allowed to stop processing a job at an arbitrary time and resume it later, possibly on a different machine. In contrast to the non-preemptive problem, Lawler and Labetoulle (1978) introduce a polynomial time algorithm to compute an optimal preemptive schedule. Thus, it is possible to design an approximation algorithm for $R||C_{\max}$ by using the value of an optimal preemptive schedule as a lower bound. Shmoys and Tardos (cited as a personal communication in Lin and Vitter 1992), show that it is possible to obtain a 4-approximation algorithm using this method. Moreover, Correa et al. (2012) prove that this is best possible by showing that the *power of preemption*, i.e., the worst case ratio of the makespan of an optimal preemptive and non-preemptive schedule, equals exactly 4.

1.1.2 Restricted assignment

The best approximation algorithm for the restricted assignment problem known so far is the $(2 - 1/m)$-approximation algorithm that follows from the general setting of $R||C_{\max}$. As mentioned above, Svensson (2012) shows how to estimate the optimal makespan within a factor $33/17 + \varepsilon$ in polynomial time. In particular, he proves that in this setting the configuration-LP has an integrality gap of at most 33/17. However, no polynomial time rounding procedure is known.

There are further results for various special cases in the restricted assignment setting, depending on the structure of the jobs and the machines, see Leung and Li (2008) for a survey. If all processing times are equal, Lin and Li (2004) prove that the restricted assignment problem is solvable in polynomial time.

1.1.3 Restricted graph balancing

The restricted graph balancing case can be interpreted as a problem on an undirected graph. The nodes of the graph correspond to machines and the edges correspond to jobs. The endpoints of an edge associated to a job j are the machines on which j has finite processing time $p_j \in \mathbb{N}^+$. 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.

Ebenlendr et al. (2008) give a 1.75-approximation algorithm based on a tighter version of the LP-relaxation by Lenstra et al. (1990). They strengthen this LP by adding inequalities that prohibit two large jobs to be simultaneously assigned to the same machine. Additionally to the 1.75-approximation algorithm for graph balancing, Ebenlendr et al. (2008) also show that it is NP-hard to approximate this problem with a better factor than 3/2. This matches the lower bound for the general problem $R||C_{\max}$. Furthermore, some special cases are studied. For example, it is known that if the underlying graph is a tree, the problem admits a PTAS. If the processing times are either 1 or 2, there is a (3/2)-approximation algorithm, which is best possible, unless $P = NP$. For these and more related results see Lee et al. (2009) and the references therein.

There is not much known for the *unrelated* graph balancing problem, where the processing time of a job can be different on its two available machines. To the best of our knowledge, everything that is known about this problem follows from results for the general case of $R||C_{\max}$. In this paper, we show that even for this special case the configuration-LP has an integrality gap of two. Hence, already for this case methods are needed which go beyond the pure configuration-LP.

1.2 The MaxMin-allocation problem

1.2.1 Unrelated machines

The MaxMin-allocation problem has drawn a lot of attention recently. For the general setting of unrelated machines, Bansal and Sviridenko (2006) show that the configuration-LP has an integrality gap of $\Omega(\sqrt{m})$. On the other hand, Asadpour and Saberi (2010) show constructively that this is tight up to logarithmic factors, yielding an algorithm with approximation ratio $O(\sqrt{m}\log^3 m)$. Relaxing the bound on the running time, Chakrabarty et al. (2009) present a polylogarithmic approximation algorithm that runs in quasi-polynomial time. In terms of complexity, the best known result is that it is NP-hard to approximate the problem within a factor of $2-\epsilon$ for any $\epsilon > 0$ (Bateni et al. 2009; Chakrabarty et al. 2009). If the number of machines is bounded by a constant, the PTAS by Lenstra et al. (1990) for a constant number of machines for $R||C_{\max}$ can easily be adapted to a PTAS for MaxMin-allocation. This is best possible since even for two machines MaxMin-allocation is NP-hard (straightforward reduction from PARTITION).

1.2.2 Restricted assignment

Bansal and Sviridenko (2006) study the case where every job has the same processing time on every machine that it can be assigned to. 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 present an algorithm with the same approximation ratio. The bound on the integrality gap is improved to $O(1)$ by Feige (2008) and to 5 and subsequently to 4 by Asadpour et al. (2008, 2012). The former proof is non-constructive using the Lovász Local Lemma, the latter two are given by a (possibly exponential time) local search algorithm. However, Haeupler et al. (2011) give a constructive version of the Lovász Local Lemma which—together with the proof by Feige (2008)—yields a polynomial time constant factor approximation algorithm.

1.2.3 Unrelated graph balancing

For the special case that every job can be assigned to at most two machines (but still with possibly different processing times on them), Bateni et al. (2009) give a 4-approximation algorithm. This is improved by Chakrabarty et al. (2009) who give an algorithm with an approximation factor of 2. Their algorithm is based on solving an LP with *knapsack-cover* inequalities, which are exponentially many. Although their algorithm does not compute a feasible solution to this LP, but rather a solution that satisfies a solution-dependent set of knapsack-cover inequalities of polynomial size, they still need to resort to the ellipsoid method in order to find such fractional solution. After, they round the fractional assignment with a procedure analogous to the one in Lenstra et al. (1990).

On the other hand, even in this special case it is NP-hard to approximate the MaxMin-allocation problem with a better ratio than 2 (Bateni et al. 2009; Chakrabarty et al. 2009). In fact, the proofs for this result use only jobs which have the same processing time on their two respective machines. Interestingly, the case that every job can be assigned to at most three machines is essentially equivalent to the general case (Bateni et al. 2009).

1.3 Our contribution

Almost all known approximation algorithms for $R||C_{\max}$ and its special cases are based on linear programs (Eben-

Table 1 The integrality gap of the configuration-LP for $R||C_{\max}$ in the various settings

	General	Unrel. graph balancing
General assignment	2	2
Restrict. assignment	$[1.5, \frac{33}{17}]$ (Ebenlendr et al. 2008; Svensson 2012)	[1.5, 1.75] (Ebenlendr et al. 2008)

lendr et al. 2008; Lenstra et al. 1990; Svensson 2012). The strongest LP that has been considered in the literature is the configuration-LP, which implicitly contains a vast class of inequalities. In this paper, we identify a surprisingly basic class of instances which captures the core complexity of the problem for LPs: the unrelated graph balancing setting. In Sect. 3, we show that even the configuration-LP has an integrality gap of 2 in the unrelated graph balancing setting and hence cannot help to improve the best known approximation factor. Interestingly, if one additionally requires that each job has the same processing time on its two machines, the integrality gap of the configuration-LP is at most 1.75 (implicitly in Ebenlendr et al. 2008). We prove our result by presenting a family of instances for which the configuration-LP has an integrality gap of 2. The instances have two novel technical properties which together lead to this large integrality gap. The first property is the usage of gadgets that we call *high-low-gadgets*. These gadgets form the seed of the inaccuracy of the configuration-LP. Second, the machines of our instances are organized in a large number of *layers*. Through the layers, the introduced inaccuracy is amplified such that the integrality gap reaches 2. To the best of our knowledge, the unrelated graph balancing case has not been considered in its own right before. Therefore, our result points to an interesting direction of future research to eventually improve the approximation factor of 2 for the general case. We note that for the restricted assignment case the configuration-LP has an integrality gap of $33/17 < 2$ (Svensson 2012). We conclude that—at least for the configuration-LP—the restricted assignment case is easier than the unrelated graph balancing case. Table 1 shows an overview of the integrality gap of the configuration-LP for the respective cases. We note that our lower bound for the unrelated graph balancing case was independently obtained by Ebenlendr et al. (2012) using a similar construction.

In Sect. 4, we study special cases for which we obtain better approximation factors than 2. In particular, we obtain a $(1 + 5/6)$-approximation algorithm for the special case of $R||C_{\max}$ where the processing times belong to a set $[\gamma, 10\gamma/3] \cup \{\infty\}$ for some $\gamma > 0$. In other words, the processing times of the jobs differ by at most a factor of 10/3. Note that the strongest known NP-hardness reductions create instances with this property. Moreover, we show that there exists a $(2 - g/p_{\max})$-approximation algorithm, where g denotes the greatest common divisor of the processing times, and $p_{\max}$ the largest finite processing time. This generalizes the result by Lin and Li (2004), that states that the case where the processing times are either 1 or infinity is polynomially solvable.

Only few approximation algorithms are known for scheduling unrelated machines which do not rely on solving a linear program. As seen above, LP-based algorithms have certain limitations and can be costly to solve. It is then preferable to have combinatorial algorithms with lower running times. For the unrelated graph balancing case of the MaxMin-allocation problem, we present an elegant combinatorial approximation algorithm with only quadratic running time and an approximation guarantee of 2. This result can be found in Sect. 5. Our algorithm uses a new preprocessing method that simplifies the complexity of a given instance and also yields a lower bound on the optimal makespan. This lower bound is as strong as the worst case bound given by the LP based on *knapsack-cover* inequalities in Chakrabarty et al. (2009). [1] Therefore we can completely avoid the use of an LP and the subsequent rounding of a fractional assignment. Indeed, we can compute directly an integral assignment by constructing a bipartite graph to link pairs of jobs which can be assigned to the same machine. A coloring for the graph then implies an assignment of the jobs to machines which ensures the claimed approximation factor.

Although the approximation guarantee is on a par with the best known algorithm—which is best possible unless $P = NP$—our approach has several advantages. The 2-approximation algorithm by Chakrabarty et al.'s (2009) resorts to the ellipsoid method, and thus has a high time complexity. On the other hand, our algorithm needs only quadratic running time. Also, it is elegant and very simple to implement. Finally, our analysis identifies the key difficulties of the problem and thus contributes to a better understanding of its underlying structure.

2 LP-based approaches

In this section, we go over the known LP-relaxations for assigning jobs to unrelated machines and elaborate on the implications of our results. In the sequel, we denote by J the set of jobs and M the set of machines of a given instance.

Canonical LP-relaxation The IP-formulation which was used by Lenstra et al. (1990) employs assignment variables $x_{i,j} \in \{0, 1\}$ that denote whether job j is assigned to machine

[1] Note that this LP has an integrality gap of exactly 2 due to the upper bound proven in Chakrabarty et al. (2009) and suitable lower bound instances, e.g., instances that stem from the NP-hardness reduction in Chakrabarty et al. (2009).

i. This formulation, which we denote by LST-IP, takes a target value for the makespan T (which will be determined later by a binary search) and does not use any objective function:

$$\text{(LST-IP)} \quad \sum_{i \in M} x_{i,j} = 1 \quad \forall j \in J, \tag{1}$$

$$\sum_{j \in J} p_{i,j} \cdot x_{i,j} \leq T \quad \forall i \in M,$$

$$x_{i,j} = 0 \quad \forall i, j : p_{i,j} > T,$$

$$x_{i,j} \in \{0, 1\} \quad \forall i \in M, j \in J.$$

The corresponding LP-relaxation of this IP, which we denote by LST-LP, can be obtained by replacing the integrality condition by $x_{i,j} \geq 0$. Let C_{LP} be the smallest integer value for T so that LST-LP is feasible, and let C^* be the optimal makespan of our instance (or equivalently, C^* is the smallest target makespan for which LST-IP is feasible). Thus, since the LP is feasible for $T = C^*$ we have that C_{LP} is a lower bound on C^*. Moreover, we can easily find C_{LP} in polynomial time with a binary search procedure.

Lenstra et al. (1990) give a rounding procedure that takes a feasible solution of LST-LP with target makespan T and returns an integral solution with makespan at most $2T$. By taking $T = C_{\mathrm{LP}} \leq C^*$ this yields a 2-approximation algorithm. The rounding, which we call *LST-rounding*, consists in interpreting the $x_{i,j}$ variables as a fractional matching in a bipartite graph, and then rounding this fractional matching to find an integral solution. This yields the following rounding theorem.

Theorem 1 (Lenstra et al. 1990) *Let $(x_{i,j})_{j \in J, i \in M}$ be a feasible solution of LST-LP with a 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 Eq. (1) and*

$$\sum_{j \in J} \bar{x}_{i,j} \cdot p_{i,j} < T + \max\{p_{i,j} : j \in J \text{ and } x_{i,j} > 0\}$$

for all $i \in M$.

We remark that in Lenstra et al. (1990) the inequality in the theorem is not strict. However, it is easy to see that the same proof yields a strict inequality. This will be useful later in Sect. 4.

Integrality Gaps and the Configuration-LP Lenstra et al. (1990) implicitly show that the LST-rounding is best possible by means of the *integrality gap* of LST-LP. For an instance I of $R||C_{\max}$, let $C_{\mathrm{LP}}(I)$ 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 $\sup_I C^*(I)/C_{\mathrm{LP}}(I)$. It is easy to see that if C_{LP} is used as a lower bound for deriving an approximation algorithm then the integrality gap is the best possible approximation guarantee that we can show. Lenstra et al. (1990) give an example showing that the integrality gap of LST-LP is arbitrarily close to 2, and thus the rounding procedure is best possible. This together with Theorem 1 implies that the integrality gap of LST-LP equals 2.

It is natural to ask whether adding a family of cuts can help to obtain a formulation with smaller integrality gap. For special cases of our problem this is indeed the case. For example, Ebenlendr et al. (2008) show that adding the inequalities

$$\sum_{j \in J : p_{i,j} > T/2} x_{i,j} \leq 1 \quad \forall i \in M, \tag{2}$$

to LST-LP yields an integrality gap of at most 1.75 in the graph balancing setting if each job has the same processing time on each of its at most two machines.

We study whether it is possible to add similar cuts to strengthen the LP for the *unrelated* graph balancing problem or even for the general case of $R||C_{\max}$. To this end we consider the *configuration-LP*, defined as follows. Let T be a target makespan, and define $\mathcal{C}_i(T)$ as the collection of all subsets of jobs whose total processing time in i is 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 variable $y_{i,C}$ for all $i \in M$ and $C \in \mathcal{C}_i(T)$, representing whether jobs in C are exactly the jobs assigned to machine i. The configuration-LP is defined as follows:

$$\sum_{C \in \mathcal{C}_i(T)} y_{i,C} = 1 \quad \forall i \in M,$$

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

$$y_{i,C} \geq 0 \quad \forall i \in M, C \in \mathcal{C}_i(T).$$

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 solve it directly in polynomial time. However, it is easy to show that the separation problem of the dual corresponds to an instance of Knapsack and thus we can solve the LP approximately in polynomial time. More precisely, given a target makespan T there is a polynomial time algorithm that either asserts that the configuration-LP is infeasible or computes a solution which uses only configurations whose makespan is at most $(1 + \varepsilon)T$, for any constant $\varepsilon > 0$ (Bansal and Sviridenko 2006). The following result, which will be proven in the next section, shows that the integrality gap of this relaxation is as large as the integrality gap of LST-LP, even for the unrelated graph balancing case.

Theorem 2 *The configuration-LP for the unrelated graph balancing problem has an integrality gap of 2.*

A solution $(y_{i,C})$ for the configuration-LP yields a feasible solution to LST-LP with the same target makespan by using the following formula

$$x_{i,j} = \sum_{C \in \mathcal{C}_i(T): j \in C} y_{i,C} \quad \forall i \in M, j \in J. \tag{3}$$

This implies that the integrality gap of the configuration-LP is not larger than the integrality gap of LST-LP, and thus it is at most 2. On the other hand, there are solutions to LST-LP that do not have corresponding 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 $T = 3/2$, it is easy to see that LST-LP is feasible, but the solution space of the configuration-LP is empty for any $T < 2$.

Now we elaborate on the relation of the two LPs, by giving a formulation in the space of the $x_{i,j}$ variables that is equivalent to the configuration-LP. Intuitively, the configuration-LP contains all possible (local) information for any single machine. Indeed, we show that any cut in the $x_{i,j}$ variables that involves only one machine is implied by the configuration-LP. Let $\alpha \in \mathbb{Q}^J$ be an arbitrary row vector. The configuration-LP will imply any cut which is of the form $\sum_{j \in J} \alpha_j x_{i,j} \le \delta_{\alpha,i}$, where $\delta_{\alpha,i}$ is properly chosen so that no single machine schedule for machine i is removed by the cut.

Proposition 1 *Fix a target makespan T. For each $\alpha \in \mathbb{Q}^J$ we define*

$$\delta_{\alpha,i} := \max\left\{\sum_{j \in S} \alpha_j : S \in \mathcal{C}_i(T)\right\}.$$

The feasibility of the configuration-LP is equivalent to the feasibility of the linear program

$$\sum_{i \in M} x_{ij} = 1 \quad \text{for all } j \in J, \tag{4}$$

$$\sum_{j \in J} \alpha_j x_{i,j} \le \delta_{\alpha,i} \quad \text{for all } \alpha \in \mathbb{Z}^J, i \in M. \tag{5}$$

Proof For a given T, let P be the polytope defined in the statement of the proposition. Consider a solution to the configuration-LP, $y_{i,C}$, and let us define

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

for all $i \in M$, $j \in J$. We show that these variables belong to P. Indeed, we first note that

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

where the last inequality follows since $y_{i,C}$ is a solution to the configuration-LP. Thus, $x_{i,j}$ satisfies the first restriction of LST-LP. Let now $\alpha \in \mathbb{Q}^J$. Then,

$$\begin{aligned}\sum_{j \in J} \alpha_j \cdot x_{i,j} &= \sum_{j \in J} \alpha_j \sum_{C \in \mathcal{C}_i(T): j \in C} y_{i,C} \\ &= \sum_{C \in \mathcal{C}_i(T)} y_{i,C} \sum_{j \in C} \alpha_j \\ &\le \max\left\{\sum_{j \in C} \alpha_j : C \in \mathcal{C}_i(T)\right\} = \delta_{\alpha,i},\end{aligned}$$

where the last inequality follows since $\sum_{C \in \mathcal{C}_i(T)} y_{i,C} = 1$ and $y_{i,C} \ge 0$ for all $i \in M, C \in \mathcal{C}_i(T)$.

For the other implication, let us now consider a solution $(x_{i,j})$ in P. We show that there exists a feasible solution to the configuration-LP. For a fixed $i \in M$, we pose the following linear program whose variables are $y_{i,C}$ for $C \in \mathcal{C}_i(T)$,

$$\sum_{C \in \mathcal{C}_i(T): j \in C} y_{i,C} = x_{i,j} \quad \text{for all } j \in J, \tag{6}$$

$$\sum_{C \in \mathcal{C}_i(T)} y_{i,C} = 1, \tag{7}$$

$$y_{i,C} \ge 0 \quad \text{for all } C. \tag{8}$$

In order to show that this system is feasible, consider a dual-variable α_j corresponding to each equality in (6), and let β be the dual-variable of equality (7). By linear duality, the previous system is feasible if and only if the following linear program does not admit a feasible solution with negative value,

$$\begin{aligned}\min \quad & \beta + \sum_{j \in J} \alpha_j x_{i,j} \\ \text{s.t.} \quad & \sum_{j \in C} \alpha_j + \beta \ge 0 \quad \text{for all } C \in \mathcal{C}_i(T).\end{aligned}$$

Consider any feasible solution β, α. We show that its value is non-negative. Without loss of generality we can assume that β takes its smaller possible value given α, i.e., $\beta = \delta_{-\alpha,i}$. Thus, Inequality (4) for vector $-\alpha$ implies that $\beta + \sum_{j \in J} \alpha_j x_{i,j} \ge 0$. We conclude that there exist variables $(y_{i,C})$ satisfying Eqs. (6)–(8) for all $i \in M$.

Now it is easy to see that these variables satisfy the restrictions of the configuration-LP. Indeed, it is enough to notice that

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

This concludes the proof of the proposition. □

As an example of the implications of this proposition, we note that adding the inequalities in (2) does not help diminishing the integrality gap of LST-LP for the unrelated graph balancing problem. This follows by taking α as the characteristic vector of the set $\{j \in J : p_{i,j} > T/2\}$ for each $i \in M$.

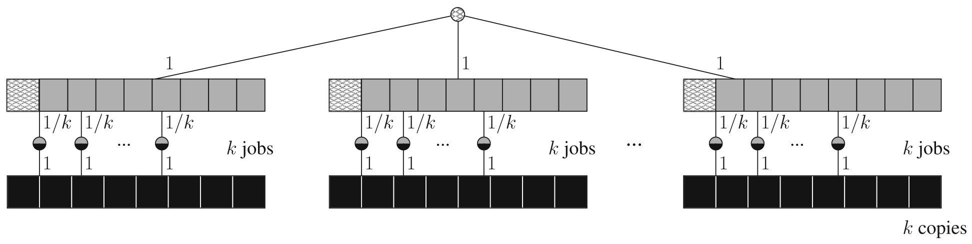

Fig. 1 A sketch of the instance of $R||C_{\max}$ where the configuration-LP has an integrality gap of $2 - \frac{1}{k}$

3 Integrality gap of 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 somehow surprising: if one additionally requires that each job has the same processing time on its two machines then Ebenlendr et al. (2008) implicitly 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. This lower bound was independently obtained by Ebenlendr et al. (2012).

3.1 Integrality gap of the configuration-LP

We describe a family of instances for the general $R||C_{max}$ problem such that for each $k \in \mathbb{N}$ there is an instance for which the configuration-LP has an integrality gap of at least $2-\frac{1}{k}$. Even though this is usually considered folklore, it will provide intuition on the behavior of the configuration-LP.

Let $k \in \mathbb{N}$. In the constructed instance there are $2k$ machines $m_1, m'_1, m_2, m'_2, \ldots, m_k, m'_k$. Then, for any pair of machines m_i, m'_i there are k jobs $j_i^1, j_i^2, \ldots, j_i^k$ which have processing time $\frac{1}{k}$ on m_i, processing time 1 on m'_i, and processing time ∞ on any other machine. Finally, there is one job j_{big} which has processing time 1 on any machine m_i and ∞ on any machine m'_i. See Fig. 1 for a sketch of this construction.

Every integral solution for this instance has a makespan of at least $2 - \frac{1}{k}$. The reason is that the job j_{big} has to be assigned to one of the machines m_i and then either m_i or m'_i has a makespan of at least $2 - \frac{1}{k}$. However, there is a solution of the configuration-LP that uses only configurations with makespan 1: we assign to every machine m_i a fraction of $\frac{1}{k}$ of the configuration $\{j_{\text{big}}\}$ and a fraction of $1 - \frac{1}{k}$ of the configuration $\{j_i^1, j_i^2, \ldots, j_i^k\}$. Also, we assign to every machine m'_i a fraction of $\frac{1}{k}$ of each configuration $\{j_i^\ell\}$ for $\ell \in \{1, \ldots, k\}$. This yields the following proposition.

Proposition 2 *The configuration-LP for $R||C_{\max}$ 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.*

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. For each integer k, we construct an instance I_k such that $p_{i,j} \in \{\frac{1}{k}, 1, \infty\}$ for each machine i and each job j. 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 requires a makespan of at least $2 - \frac{1}{k}$.

Let $k \in \mathbb{N}$ and let N be the smallest integer satisfying $k^N/(k-1)^{N+1} \geq \frac{1}{2}$. Consider a k-ary tree of height $N - 1$, i.e., a tree of height $N - 1$ in which apart from the leaves every vertex has k children. Let r be the root of the tree. We say that all vertices with the same distance to the root are in the same *layer*. Let L be the set containing the leaves. For every leaf $v \in L$, we introduce another vertex $w(v)$ and k edges between v and $w(v)$. (Hence, $v \in L$ is no longer a leaf.) We call such a pair of vertices $v, w(v)$ a *high-low-gadget*. Observe that the resulting "tree" has height N, i.e., the distance of any vertex to r is at most N.

Based on this, we describe our instance of unrelated graph balancing. For each vertex v of the constructed graph we introduce a machine m_v. For each edge $e = \{u, v\}$ we introduce a job j_e. Assume that u is closer to r 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 infinite processing time on any other machine. This motivates the term "high-low-gadget" of a pair $v, w(v)$ for $v \in L$: each job inside such a gadget has a high processing time on $m_{w(v)}$ and a low processing time on m_v. We now make a copy of the whole construction, obtaining two identical graphs and corresponding scheduling instances. Let r_1 and r_2 be the roots of the each of the constructed graphs, respectively. Similarly as before, for $i \in \{1, 2\}$ we define L_i as the set of all vertices whose distance to r_i equals $N-1$. Let

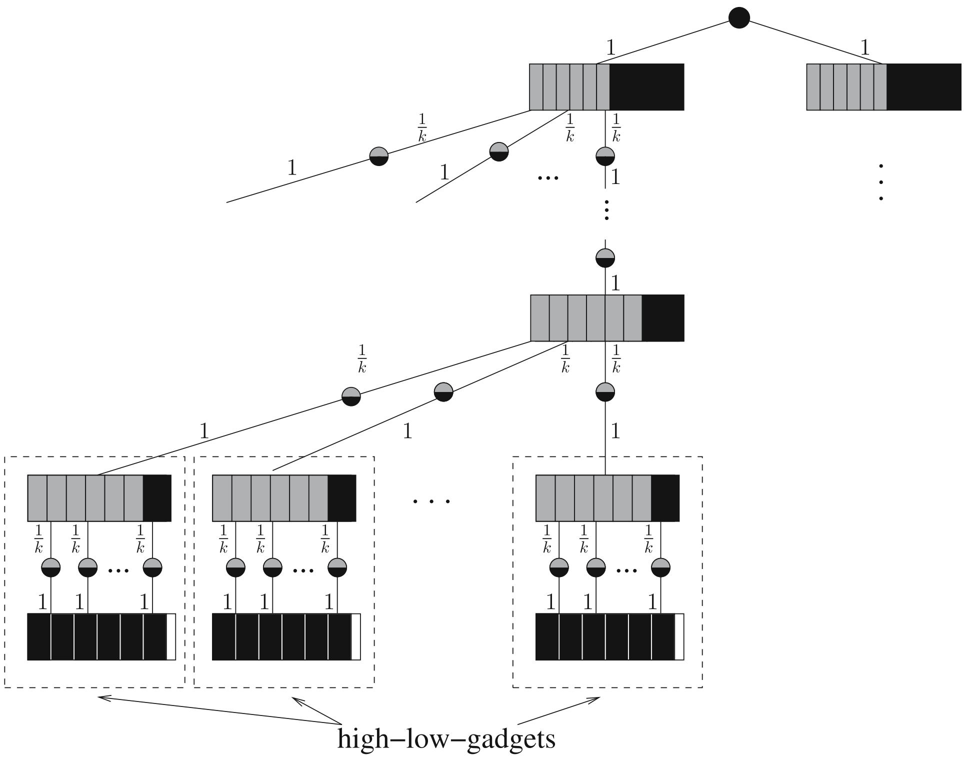

Fig. 2 A sketch of the construction for the instance of unrelated graph balancing with an integrality gap of $2 - O\left(\frac{1}{k}\right)$. The jobs on the machines correspond to the fractional solution of the configuration-LP for this instance with makespan $T = 1 + \frac{1}{k}$

$m_r^{(1)}$ 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 $m_r^{(2)}$, and ∞ on any other machine. Denote by I_k the resulting instance. See Fig. 2 for a sketch.

To gain some intuition for the construction, consider a high-low-gadget consisting of two machines m_v, $m_{w(v)}$ for some $v \in L_1 \cup L_2$. In any solution with a makespan of at most $1 + \frac{1}{k}$, it is clear that m_v can schedule only the jobs whose respective edges connect v and $w(v)$. However, we will see in the sequel that there are solutions for the configuration-LP with makespan $1 + \frac{1}{k}$ in which also a fraction of the job with processing time 1 is scheduled on m_v (similarly to the construction presented in Sect. 3.1). Since we chose a large number of layers, this fraction will be amplified through the layers to the root until we obtain a feasible solution to the configuration-LP using only configurations with makespan at most $1+\frac{1}{k}$. However, any integral solution has a makespan of at least $2-\frac{1}{k}$ as we will prove in the following lemma. This implies that the configuration-LP has an integrality gap of 2.

Lemma 1 *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 makespan strictly smaller than 2. W.l.o.g. assume that j_{big} is assigned to machine $m_r^{(1)}$. As 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_1 of the first tree such that j_e is *not* assigned to $m_r^{(1)}$. 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 $v \in L_1$ 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 $m_{w(v)}$. 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 of the configuration-LP for I_k which uses only configurations with makespan $1+\frac{1}{k}$. To this end, we introduce the concept of *$j-\alpha$-solutions* for the configuration-LP. A $j-\alpha$-solution is a solution for the configuration-LP whose right-hand side is modified as follows: job j does not need to be fully assigned but only to an extent of a fraction $\alpha \leq 1$. This value α corresponds to the fraction of the big job assigned to a machine like m_v as described above.

For any $h \in \mathbb{N}$ denote by $I_k^{(h)}$ a subinstance of I_k defined as follows: Take a vertex v whose distance to r_1 equals to $N - h$, and consider the subtree $T(v)$ rooted at v. That is, $T(v)$ is the set of all vertices whose shortest path to r_1 passes through v. Note that h can be interpreted as the height of v in the corresponding tree. For the subinstance $I_k^{(h)}$, we take all machines and jobs which correspond to vertices and edges in $T(v)$. We remark that since our construction is symmetric it does not matter which vertex v of distance $N - h$ to

r_1 (or r_2) 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 $j^{(h)} - \alpha^{(h)}$-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. It explains the inaccuracy of the configuration-LP introduced by the high-low-gadgets.

Lemma 2 *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 $v \in L_1 \cup L_2$ be the vertex which corresponds to the root of $I_k^{(1)}$. Let $j_1^{(0)}, \ldots, j_k^{(0)}$ be the jobs which have processing time 1 on $m_{w(v)}$ and processing time $\frac{1}{k}$ on m_v.

For $m_{w(v)}$ the configurations with a makespan of at most $1 + \frac{1}{k}$ are $C_\ell := \left\{j_\ell^{(0)}\right\}$ for each $\ell \in \{1, \ldots, k\}$ (so only the configurations which consist of exactly one job having processing time 1 on $m_{w(v)}$). Then we define $y_{m_{w(v)},C_\ell} := \frac{1}{k}$ for each ℓ. Hence, for each job $j_\ell^{(0)}$ a fraction of $\frac{k-1}{k}$ remains unassigned. For m_v there are the following (maximal) configurations: $C_{\text{small}} := \left\{j_1^{(0)}, \ldots, j_k^{(0)}\right\}$ (so the set of all jobs having processing time $\frac{1}{k}$ on m_v) and $C_{\text{big}}^\ell := \left\{j^{(1)}, j_\ell^{(0)}\right\}$ for each $\ell \in \{1, \ldots, k\}$ (the single job with processing 1 on m_v together with one of the jobs with processing time $\frac{1}{k}$ on m_v). We define $y_{m_v,C_{\text{big}}^\ell} := \frac{1}{k(k-1)}$ for each ℓ and $y_{m_v,C_{\text{small}}} := 1 - \frac{1}{k-1}$. This assigns each job $j_\ell^{(0)}$ completely and the job $j^{(1)}$ to an extent 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. It shows how the value α of our $j - \alpha$-solutions is increased by the layers of our construction, and thus the effect of the high-low-gadgets is amplified.

Lemma 3 *Assume that we are given a $j^{(n)}$–$\left(k^n/(k-1)^{n+1}\right)$-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 $j^{(n+1)}$- $\left(k^{n+1}/(k-1)^{n+2}\right)$-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 $\ell \in \{1, \ldots, k\}$ let $j_\ell^{(n)}$ be the jobs which have processing time $\frac{1}{k}$ on m_v.

Inside 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 extent of $\left(k^n/(k-1)^{n+1}\right)$. Like in Lemma 2 the (maximal) configurations for m_v are given by $C_{\text{small}} := \left\{j_1^{(n)}, \ldots, j_k^{(n)}\right\}$ and $C_{\text{big}}^\ell := \left\{j^{(n+1)}, j_\ell^{(n)}\right\}$ for each $\ell \in \{1, \ldots, k\}$. We define the value $y_{m_v,C_{\text{big}}^\ell} := k^n/(k-1)^{n+2}$ for each ℓ and $y_{m_v,C_{\text{small}}} := 1 - k^{n+1}/(k-1)^{n+2}$. This assigns each job $j_\ell^{(n)}$ completely and the job $j^{(n+1)}$ is assigned to an extent of $k \cdot k^n/(k-1)^{n+2} = k^{n+1}/(k-1)^{n+2}$. □

Now our main theorem follows from the previous lemmas.

Theorem 2 *The configuration-LP for the unrelated graph balancing problem has an integrality gap of 2.*

Proof Lemmas 1, 2, and 3 imply that for 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. The upper bound of 2 follows from Lenstra et al. (1990). □

4 Cases with better approximation factors

In this section, we identify classes of instances of $R||C_{\max}$ 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 Bounded range of processing times

We show that if the finite processing times of the jobs differ by at most a factor of 10/3 we can give a $(1 + \frac{5}{6})$-approximation algorithm. Hence, using reductions of this type (which applies to the strongest known NP-hardness reductions for $R||C_{\max}$) one cannot rule out a $(2-\epsilon)$-approximation algorithm.

Theorem 3 *Consider instances of $R||C_{\max}$ with a value γ such that $p_{i,j} \in \left[\gamma, 10\gamma/3\right] \cup \{\infty\}$ for all machines i and all jobs j. For these instances there is a $1 + \frac{5}{6} \approx 1.83$-approximation algorithm.*

Proof We use the LST-LP and a combinatorial algorithm, depending on the target makespan T given by the binary search. Assume we are given a target makespan T. If $T \geq 4\gamma$ then we solve the (original) LST-LP. If it is feasible, due to Theorem 1 we know that we can round it to an integral solution whose makespan is bounded by

$$T + \frac{10}{3}\gamma \leq \left(1 + \frac{5}{6}\right) T.$$

So now assume that $T < 4\gamma$. For that case we give a matching-based combinatorial algorithm producing a solution with makespan at most $\left(1+\frac{5}{6}\right)T$. We introduce a bipartite graph with one vertex v_j for each job $j \in J$ and three vertices $w_{i,1}, w_{i,2}, w_{i,3}$ for each machine $i \in M$. We introduce an edge between a vertex v_j and a vertex $w_{i,\ell}$ if and only if $p_{i,j} \leq T/\ell$ for each job j, each machine i, and each $\ell \in \{1, 2, 3\}$. We see that in any feasible solution with makespan at most T, each machine i has at most one job j with $p_{i,j} > T/2$, at most two jobs j with $p_{i,j} > T/3$ and at most three jobs in total. Hence, if there is a solution with makespan at most T then there is a (perfect) matching in which each vertex v_j is matched. On the other hand, each perfect matching induces a schedule in which the makespan of each machine is bounded by $T+\frac{T}{2}+\frac{T}{3}=\left(1+\frac{5}{6}\right)T$. We compute a maximum matching. If each vertex v_j is matched, this induces us a solution with makespan at most $\left(1+\frac{5}{6}\right)T$. On the other hand, if no such matching exists, we output that the optimum is stricly larger than T and continue the binary search procedure on T. □

Unfortunately, we do not gain anything by generalizing this method further to, e.g., the case that $p_{i,j} \in [\gamma, 4\gamma] \cup \{\infty\}$. The reason is that $T+\frac{T}{2}+\frac{T}{3}=T(1+\frac{5}{6}) < 2T$ but $T+\frac{T}{2}+\frac{T}{3}+\frac{T}{4} \approx 2.08T > 2T$ and a 2-approximation algorithm is already known.

4.2 Bounded GCD of processing times

The inapproximability results for $R||C_{\max}$ given in Ebenlendr et al. (2008) and Lenstra et al. (1990) use only jobs j such that $p_{i,j} \in \{1, 2, 3, \infty\}$ for all machines i. We show now that for classes of instances which use only a finite set of processing times, there exists an approximation algorithm with a performance guarantee strictly better than 2.

Theorem 4 *There exists a $(2-\alpha)$-approximation algorithm for the problem of minimizing makespan on unrelated machines, where*

$$\alpha = \frac{\gcd\{p_{i,j} \mid i \in M, j \in J, p_{i,j} < \infty\}}{\max\{p_{i,j} \mid i \in M, j \in J, p_{i,j} < \infty\}}.$$

Proof We give a slighty strengthened analysis of the 2-approximation algorithm by Lenstra et al. (1990). Let

$$g := \gcd\{p_{i,j} | i \in M, j \in J, p_{i,j} < \infty\}$$

and

$$P := \max\{p_{i,j} | i \in M, j \in J, p_{i,j} < \infty\}.$$

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 (can be computed by a binary search). Assume we have computed a fractional solution for LST-LP with target makespan T^*. We apply LST-rounding to this fractional solution. By Theorem 1, the makespan of the rounded solution is *strictly* less than $T^* + P'$ with $P' = \max\{p_{i,j} : j \in J, p_{i,j} \leq T^*\}$. Since the obtained makespan, P', and T^* are multiples of g, we conclude that the former is bounded by $T^* + P' - g$. Let β be a non-negative integer such that $T^* = P' + \beta \cdot g$. The following calculation then shows the claimed approximation guarantee:

$$\begin{aligned} T^* + P' - g &\leq T^*\left(2 - \frac{(\beta+1)g}{T^*}\right) \\ &= T^*\left(2 - \frac{(\beta+1)g}{P' + \beta \cdot g}\right) \\ &\leq T^*(2-\alpha)\,. \end{aligned}$$

□

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 1 *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 processing time $p_{i,j}$ arising in I it holds that $p_{i,j} \in P \cup \{\infty\}$. Then for the family of instances $\mathcal{I}$ there is an approximation algorithm with performance guarantee $2 - \alpha$ with $\alpha = \gcd\{p|p \in P\}/\max\{p|p \in P\}$.*

5 MaxMin-allocation problem

In this section, we study the MaxMin-balancing problem on unrelated machines. Recall that for this problem the objective is to maximize the minimum load of the machines and every job can be assigned to at most two machines (with possibly different processing times on each machine). The best known approximation algorithm for this problem is a 2-approximation algorithm based on a linear program which uses knapsack-cover inequalities (Chakrabarty et al. 2009). This yields the best possible approximation ratio unless $P = NP$. However, we show that it is in fact not necessary to solve a linear program (with the computationally costly ellipsoid method) to achieve this factor. Instead, we present here a purely combinatorial 2-approximation algorithm with only quadratic running time which is quite easy to implement.

5.1 A 2-approximation 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 $J_i = \{j_{i,1}, j_{i,2}, \ldots\}$ 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 $A_i = \{a_{i,1}, a_{i,2}, \ldots\}$ 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 (or one of those jobs in case there is a tie). For any set of jobs J'_i and a machine i we define $p(J'_i) := \sum_{j \in J'_i} p_{i,j}$.

Denote by $p_{i,\ell}$ the processing time of job $a_{i,\ell}$ on machine i. We assume that the elements of A_i 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 4 *If during the preassignment phase the algorithm outputs 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 $k \geq 0$. The reason for the latter edges is that later exactly one of the two jobs $j_{i,2k+1}$, $j_{i,2k+2}$ will be assigned to i.

Lemma 5 *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. On a cycle, the edges of these two types alternate and hence the graph is bipartite. □

Due to Lemma 5 we can color G with two colors, black and white. Let i be a machine. We assign each 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 6 *The algorithm outputs a solution whose value is at least T/2.*

Proof Let i be a machine. We show that the total processing time 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 processing time 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 an algorithm for the entire problem an additional binary search is necessary to find the correct value of T. Now we discuss how to implement the overall algorithm efficiently.

First, we test whether $n < m$. If this is the case then any (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 processing time (in the list that we sort we have two entries for every job, each corresponding to one of its possible processing 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 also requires only linear time.

For the binary search, we need to try at most $\log D$ values, where D is defined by $D := \sum_{i,j} p_{i,j}$. 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 needs time $O(|I| \log |I|)$. For every value T that we try, $O(|I|)$ steps are necessary. This yields an overall running time of $O(|I|^2)$.

Theorem 5 *There is a 2-approximation algorithm for the MaxMin-balancing problem with running time* $O(|I|^2)$.

6 Conclusion

As discussed above, the problem of minimizing the makespan on unrelated machines is one of the most prominent open problems in scheduling. To close the gap between the 2-approximation algorithm by Lenstra et al. (1990) and their 3/2-hardness result seems a very challenging task. Since the machines are unrelated, usual approaches like for identical machines cannot be used (if the number of machines is part of the input). However, our results show that most LP-based approaches are deemed to fail, even for the unrelated graph balancing case. Hence, when trying to find a better approximation algorithm it seems reasonable to study the latter setting. To the best of our knowledge, it has not been considered in its own right so far.

In the paper by Ebenlendr et al. (2008), the setting of graph balancing *and* restricted assignment is studied. Our results and the work by Svensson (2012) indicate that the restricted assignment feature is actually the reason why this improvement was possible, rather than the restriction to the graph balancing case. In (2012), Svensson proves an upper bound for the integrality gap of the configuration-LP of $33/17$ in the restricted assignment case. To the best of our knowledge, for the restricted assignment case no instance is known for which the configuration-LP has an integrality gap larger than $3/2$. It would be interesting to construct such an instance. In fact, in our constructions we used only the processing times $\{\epsilon, 1, \infty\}$. It is not clear to us how more processing times in an instance could help to show a larger integrality gap.

For the MaxMin-allocation problem, the algorithm presented in Sect. 5 achieves the best known approximation factor in its setting (and it is in fact best possible, unless $P = NP$). To the best of our knowledge, it is the only such algorithm for a non-trivial case of the MaxMin-allocation problem which does not rely on solving a linear program, in particular not the computationally expensive configuration-LP. It would be interesting whether purely combinatorial algorithms are also possible for other settings of the problem.

Acknowledgments This work was partially supported by Berlin Mathematical School (BMS), by the DFG Focus Program 1307 within the project "Algorithm Engineering for Real-time Scheduling and Routing," by FONDECYT project 3130407, and by Nucleo Milenio Información y Coordinación en Redes ICM/FIC P10-024F.

References

Asadpour, A., Feige, U., & Saberi, A. (2008). Santa Claus meets hypergraph matchings. In *Proceedings of the 11th International Workshop and 12th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX-RANDOM 2008)*. LNCS (Vol. 5171, pp. 10–20). Berlin: Springer.

Asadpour, A., Feige, U., & Saberi, A. (2012). Santa Claus meets hypergraph matchings. *ACM Transactions on Algorithms*, *8*, Art. No. 24.

Asadpour, A., & Saberi, A. (2010). An approximation algorithm for max-min fair allocation of indivisible goods. *SIAM Journal on Computing*, *39*, 2970–2989.

Bansal, N., & Sviridenko, M. (2006). The Santa Claus problem. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC 2006)* (pp. 31–40).

Bateni, M., Charikar, M., & Guruswami, V. (2009). Maxmin allocation via degree lower-bounded arborescences. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC 2009)* (pp. 543–552).

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

Correa, J. R., Skutella, M., & Verschae, J. (2012). The power of preemption on unrelated machines and applications to scheduling orders. *Mathematics of Operations Research*, *37*, 379–398.

Ebenlendr, T., Krčál, M., & Sgall, J. (2008). Graph balancing: A special case of scheduling unrelated parallel machines. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008)* (pp. 483–490).

Ebenlendr, T., Krčál, M., & Sgall, J. (2012). Graph balancing: A special case of scheduling unrelated parallel machines. Algorithmica 1–19. doi:10.1007/s00453-012-9668-9.

Feige, U. (2008). On allocations that maximize fairness. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008)* (pp. 287–293).

Gairing, M., Monien, B., & Woclaw, A. (2007). A faster combinatorial approximation algorithm for scheduling unrelated parallel machines. *Theoretical Computer Science*, *380*, 87–99.

Haeupler, B., Saha, B., & Srinivasan, A. (2011). New constructive aspects of the Lovász local lemma. *Journal of the ACM*, *58*, Art. No. 28.

Horowitz, E., & Sahni, S. (April 1976). Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, *23*, 317–327.

Karmarkar, N., & Karp, R. M. (1982). An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 1982)* (pp. 312–320).

Lawler, E. L., & Labetoulle, J. (1978). On preemptive scheduling of unrelated parallel processors by linear programming. *Journal of the ACM*, *25*, 612–619.

Lee, K., Leung, J. Y., & Pinedo, M. L. (2009). A note on graph balancing problems with restrictions. *Information Processing Letters*, *110*, 24–29.

Lenstra, J. K., Shmoys, D. B., & Tardos, E. (1990). Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, *46*, 259–271.

Leung, J. Y., & Li, C. (2008). Scheduling with processing set restrictions: A survey. *International Journal of Production Economics*, *116*, 251–262.

Lin, J.-H., & Vitter, J. S. (1992). epsilon-Approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC 1992)* (pp. 771–782).

Lin, Y., & Li, W. (2004). Parallel machine scheduling of machine-dependent jobs with unit-length. *European Journal of Operational Research*, *156*, 261–266.

Schuurman, P., & Woeginger, G. J. (1999). Polynomial time approximation algorithms for machine scheduling: Ten open problems. *Journal of Scheduling*, *2*, 203–213.

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

Svensson, O. (2012). Santa Claus schedules jobs on unrelated machines. *SIAM Journal on Computing*, *41*, 1318–1341.

Verschae, J., & Wiese, A. (2011). On the configuration-LP for scheduling on unrelated machines. In *Proceedings of the 19th European Symposium on Algorithms (ESA 2011)*. Lecture Notes in Computer Science (Vol. 6942, pp. 530–542). Berlin: Springer.