Inapproximability Results for Scheduling with Interval and Resource Restrictions

Marten Maack
Department of Computer Science, Kiel University, Kiel, Germany
mmaa@informatik.uni-kiel.de

Klaus Jansen
Department of Computer Science, Kiel University, Kiel, Germany
kj@informatik.uni-kiel.de

Abstract

In the restricted assignment problem, the input consists of a set of machines and a set of jobs each with a processing time and a subset of eligible machines. The goal is to find an assignment of the jobs to the machines minimizing the makespan, that is, the maximum summed up processing time any machine receives. Herein, jobs should only be assigned to those machines on which they are eligible. It is well-known that there is no polynomial time approximation algorithm with an approximation guarantee of less than 1.5 for the restricted assignment problem unless P=NP. In this work, we show hardness results for variants of the restricted assignment problem with particular types of restrictions.

For the case of interval restrictions – where the machines can be totally ordered such that jobs are eligible on consecutive machines – we show that there is no polynomial time approximation scheme (PTAS) unless P=NP. The question of whether a PTAS for this variant exists was stated as an open problem before, and PTAS results for special cases of this variant are known.

Furthermore, we consider a variant with resource restriction where the sets of eligible machines are of the following form: There is a fixed number of (renewable) resources, each machine has a capacity, and each job a demand for each resource. A job is eligible on a machine if its demand is at most as big as the capacity of the machine for each resource. For one resource, this problem has been intensively studied under several different names and is known to admit a PTAS, and for two resources the variant with interval restrictions is contained as a special case. Moreover, the version with multiple resources is closely related to makespan minimization on parallel machines with a low rank processing time matrix. We show that there is no polynomial time approximation algorithm with a rate smaller than $48/47 \approx 1.02$ or 1.5 for scheduling with resource restrictions with 2 or 4 resources, respectively, unless P=NP. All our results can be extended to the so called Santa Claus variants of the problems where the goal is to maximize the minimal processing time any machine receives.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Scheduling algorithms

Keywords and phrases Scheduling, Restricted Assignment, Approximation, Inapproximability, PTAS

Digital Object Identifier 10.4230/LIPIcs.STACS.2020.5

Related Version A full version of the paper is available at `http://arxiv.org/abs/1907.03526`.

Funding *Marten Maack*: German Academic Exchange Service (DAAD) scholarship
Klaus Jansen: German Research Foundation (DFG) project JA 612/15-2

Acknowledgements We thank Malin Rau and Lars Rohwedder for helpfull discussions on the problem.

1 Introduction

Consider the restricted assignment problem: Given a set of machines $\mathcal{M}$ and a set of jobs $\mathcal{J}$ with a processing time or size p_j and a subset of eligible machines $\mathcal{M}(j) \subseteq \mathcal{M}$ for each job j, the goal is to find a schedule $\sigma : \mathcal{J} \to \mathcal{M}$ with $\sigma(j) \in \mathcal{M}(j)$ for each job j and minimizing the makespan $C_{\max}(\sigma) = \max_{i \in \mathcal{M}} \sum_{j \in \sigma^{-1}(i)} p_j$.

In a seminal work, Lenstra, Shmoys and Tardos [19] presented a 2-approximation for restricted assignment and also showed that there is no polynomial time approximation algorithm with rate smaller than 1.5 for the problem, unless P=NP. Closing this gap is a prominent open problem in approximation and scheduling theory [26, 33]. If there are no restrictions, i.e., $\mathcal{M}(j) = \mathcal{M}$ for each job j, we have the classical problem of makespan minimization on identical parallel machines (machine scheduling) which is already strongly NP-hard. On the other hand, machine scheduling is well-known to admit a polynomial time approximation scheme (PTAS) due to a classical result by Hochbaum and Shmoys [10]. In recent years, the approximability of special cases of restricted assignment has been intensively studied (see, e.g., [3, 7, 12, 14]) with one line of research focusing on the existence of approximation schemes (see, e.g., [8, 13, 23, 24]). The present work seeks to contribute in this research direction. Omitted details and proofs can be found in the long version of the paper [22].

Interval Restrictions. Arguably one of the most natural variants of the restricted assignment problem is the case of scheduling with interval restrictions (RAI). In this variant, the machines are totally ordered and each job is eligible on consecutive machines. More precisely, we have $\mathcal{M} = \{M_1, \ldots, M_m\}$, and for each job j we have some indices $\ell, r \in [m]$ such that $\mathcal{M}(j) = \{M_\ell, \ldots, M_r\}$. Several special cases of RAI are known to admit a PTAS: the hierarchical case [24], where for each job the interval of eligible machines starts with the first machine; the nested case [23, 8], where $\mathcal{M}(j) \subseteq \mathcal{M}(j')$, $\mathcal{M}(j') \subseteq \mathcal{M}(j)$ or $\mathcal{M}(j) \cap \mathcal{M}(j') = \emptyset$ for each pair of jobs (j, j'); and the inclusion-free case [27, 17], where $\mathcal{M}(j) \subseteq \mathcal{M}(j')$ implies that j and j' share either their first or last eligible machine. Furthermore, for general RAI, a $2 - 2/(\max_{j \in \mathcal{J}} p_j)$-approximation due to Schwarz [27] is known (assuming integral processing times); and the special case with two distinct processing times is even polynomial time solvable [32]. Note that the problem has also been studied in the context of online algorithms (see [18, 21]).

The question of whether there is a PTAS for RAI has been posed by several authors [15, 27, 32]. As the main result of the present work, we resolve this question in the negative:

▶ **Theorem 1.** *There is no PTAS for scheduling with interval restrictions unless P=NP.* [1]

We prove this theorem in Section 2.

Resource Restrictions. The second variant considered in this work, is the problem of scheduling with resource restrictions with R resources (RAR(R)). Herein, a set $\mathcal{R}$ of R (renewable) resources is given, each machine i has a resource capacity $c_r(i)$ and each job j has a resource demand $d_r(j)$ for each $r \in \mathcal{R}$. Job j is eligible on machine i if $d_r(j) \leq c_r(i)$ for each resource r. For $R = 1$, the problem is equivalent to the mentioned hierarchical case and

[1] There is a paper [16] claiming to have found a PTAS for RAI. However, according to [29], the result is not correct and the authors published a revised version of the paper [17] claiming a less general result, namely, a PTAS for the inclusion-free case.

has been studied intensively [20, 21]. Furthermore, it is not hard to see that RAI is properly placed between RAR(1) and RAR(2) and hence there is a close relationship between the two problems. For arbitrary R, the problem was mentioned in a work by Bhaskara et el. [1] under the name of geometrically restricted scheduling[2] but to the best of our knowledge it has not been further studied up to now. There is, however, a close relationship to the low rank version of makespan minimization on unrelated parallel machines (unrelated scheduling) introduced in [1]. For scheduling with resource restrictions, we show:

▶ **Theorem 2.** *There is no approximation algorithm with rate less than* $48/47 \approx 1.02$ *or* 1.5 *for scheduling with resource restrictions with* 2 *or* 4 *resources, respectively, unless P=NP.*

We prove this theorem in Section 3. In the long version of the paper [22], we provide more details concerning RAR(R), e.g., a comparative analysis of RAR(R), RAI and low rank unrelated scheduling.

Santa Claus. The problems of restricted assignment and unrelated scheduling are also studied with the reverse objective of maximizing the minimal machine load $\min_{i \in \mathcal{M}} \sum_{j \in \sigma^{-1}(i)} p_{ij}$. Machine scheduling problems with this objective are sometimes called the Santa Claus version of the respective problem. We remark that all our results can be transferred to the Santa Claus versions of the considered problems in a straight-forward fashion.

Further Related Work. First note that if the number of machines is constant, there is a fully polynomial time approximation scheme (FPTAS) already for unrelated scheduling [11]. Furthermore, for some broad overview concerning parallel machine scheduling with different kinds of restrictions in the context of online and approximation algorithms, we refer to the surveys by Lee et al. [18] and Leung and Li [20, 21].

We already discussed many variants of restricted assignment that admit a PTAS. In particular, Ou, Leung and Li [24] presented a PTAS for the hierarchical case; Epstein and Levin [8] and Muratore, Schwarz and Woeginger [23] for the nested case; and Schwarz [27] and Khodamoradi et al. [17] for the inclusion-free case. Another case that has been studied in the literature is the tree-hierarchical case, where the machines can be arranged in a rooted tree such that for each job the set of eligible machines corresponds to a path starting at the root. It was shown to admit a PTAS by Epstein and Levin [8] and Schwarz [28]. It is not hard to see that all of the above cases contain the hierarchical case as a subcase, and that the tree-hierarchical, nested and inclusion-free case are distinct. There is, however, a variant admitting a PTAS that covers both the nested and the tree-hierarchical case: For each instance of the restricted assignment problem the corresponding incidence graph is a bipartite graph whose nodes are given by the jobs and machines and a job j is adjacent to a machine i if j is eligible on i. Jansen, Maack and Solis-Oba [13] showed that there is PTAS for restricted assignment for the case that the clique- or rank-width of the incidence graph is constant. Furthermore, if the incidence graph is a bi-cograph the clique-width is well-known to be small and this case covers the nested and tree-hierarchical case. The inclusion-free case, on the other hand, is equivalent to the case that the incidence graph is a bipartite permutation graph [17] which does not have a bounded clique-width [2]. Note that RAR(1) or RAI are equivalent to the cases that the incidence graph is a chain [9] or convex graph [16], respectively.

[2] The demands $d(j)$ and capacities $c(i)$ may be interpreted as points in R-dimensional space.

Preliminaries. We consider polynomial time approximation algorithms: Given an instance I of an optimization problem, an α-approximation A for this problem produces a solution in time poly($|I|$), where $|I|$ denotes the input length. For the objective function value $A(I)$ of this solution it is guaranteed that $A(I) \leq \alpha \textsc{opt}(I)$, in the case of an minimization problem, or $A(I) \geq (1/\alpha)\textsc{opt}(I)$, in the case of an maximization problem, where $\textsc{opt}(I)$ is the value of an optimal solution. We call α the *approximation guarantee* or *rate* of the algorithm. In some cases a polynomial time approximation scheme (PTAS) can be achieved, that is, an $(1+\varepsilon)$-approximation for each $\varepsilon > 0$. If for such a family of algorithms the running time is polynomial in both $1/\varepsilon$ and $|I|$ it is called *fully polynomial* (FPTAS).

Nearly all the reductions in this work follow the same pattern: Given an instance I of the starting problem, we construct an instance I' of the variant of the restricted assignment problem considered in the respective case. For I', all job sizes are integral and upper bounded by some constant T such that the overall size of the jobs equals $|\mathcal{M}|T$. Obviously, if for such an instance a machine receives jobs with overall size more or *less* than T, the makespan of the schedule is greater than T. Then we show that that there exists a schedule with makespan T for I', if and only if I is a yes-instance. This rules out the existence of an approximation algorithm with rate smaller than $(T+1)/T$ (or $T/(T-1)$ for the Santa Claus version) and a PTAS in particular.

2 Interval Restrictions

The sole goal of this section is to prove Theorem 1, that is, the non-existence of a PTAS for RAI (given P≠NP). Our starting point for the reduction is a satisfiability problem 3-SAT* that we tailor to our needs. We show that 3-SAT* is NP-hard via a straight forward reduction from the 1-in-3-SAT problem, which is well-know to be NP-complete [25] and discussed in more detail below. Next, we provide a reduction from 3-SAT* to the classical restricted assignment problem (with arbitrary sets of eligible machines). This reduction introduces some of the needed gadgets and ideas for the main result which is discussed in detail thereafter.

Starting Point. An instance of 1-in-3-SAT is a conjunction of clauses with 3 literals each. Each clause is a formula depending on 3 literals that is satisfied if and only if exactly one of its literals takes the value $\top$. We call such formulas 1-in-3-clauses in the following and define 2-in-3-clauses correspondingly. Note that we denote the truth values "true" and "false" by $\top$ and $\bot$ in the following. An instance of the problem 3-SAT* also is a conjunction of clauses with exactly 3 literals each. However, each of the clauses is either a 1-in-3-clause or a 2-in-3-clause and there are as many clauses of the first as of the second type. Furthermore, we require that each literal occurs *exactly twice*. In the following, we denote a 1-in-3-clause or 2-in-3-clause with literals z_1, z_2 and z_3 by $(z_1, z_2, z_3)_1$ or $(z_1, z_2, z_3)_2$, respectively.

To see that 3-SAT* is NP-hard, consider an instance of 1-in-3-SAT with n variables $x_1, \dots, x_n$ and m clauses. We now construct an equivalent 3-SAT* instance. Let d_i be the number of times the variable x_i occurs in the given 1-in-3-SAT formula. For each variable x_i, we introduce new variables $x_{i,1}, \dots, x_{i,d_i}$ and $y_{i,1}, \dots, y_{i,d_i}$ along with clauses $(x_{i,1}, \neg x_{i,2}, y_{i,1})_2, \dots, (x_{i,d_i-1}, \neg x_{i,d_i}, y_{i,d_i-1})_2$, $(x_{i,d_i}, \neg x_{i,1}, y_{i,d_i})_2$ and clauses $(y_{i,j}, \neg y_{i,j}, \neg y_{i,j})_1$ for each $j \in [d_i]$. Note that each variable $y_{i,j}$ has to take the value $\top$ in a satisfying assignment, due to the clause $(y_{i,j}, \neg y_{i,j}, \neg y_{i,j})_1$. The remaining clauses ensure, that for each i the variables $x_{i,1}, \dots x_{i,d_i}$ have the same value in a satisfying assignment. Furthermore, for each of the clauses of the original problem, we introduce one 1-in-3-clause and one 2-in-3-clause. The 1-in-3-clauses are obtained by exchanging the j-th occurrence of each variable x_i with $x_{i,j}$.

Table 1 The sizes and sets of eligible machines of the jobs in the simple reduction. The entry for $\texttt{CMach}_{i,s}$ marks the private load of the machine. The target makespan is given by $T = 322$.

Job	Size	Eligible Machines
$\texttt{CMach}_{i,s}$	111	$\texttt{CMach}_{i,s}$
$\texttt{CJob}^{\top}_{i,s'}$	100	$\texttt{CMach}_{i,1}$, $\texttt{CMach}_{i,2}$, $\texttt{CMach}_{i,3}$
$\texttt{CJob}^{\bot}_{i,s'}$	101	$\texttt{CMach}_{i,1}$, $\texttt{CMach}_{i,2}$, $\texttt{CMach}_{i,3}$
$\texttt{TJob}^{\top}_{j}$	100	$\texttt{TMach}_{j,1}$, $\texttt{TMach}_{j,2}$
$\texttt{TJob}^{\bot}_{j}$	102	$\texttt{TMach}_{j,1}$, $\texttt{TMach}_{j,2}$
$\texttt{VJob}^{\top}_{j,t}$	111	$\texttt{TMach}_{j,\lceil t/2 \rceil}$, $\texttt{CMach}_{\kappa(j,t)}$
$\texttt{VJob}^{\bot}_{j,t}$	110	$\texttt{TMach}_{j,\lceil t/2 \rceil}$, $\texttt{CMach}_{\kappa(j,t)}$

Moreover, the 2-in-3-clauses are obtained by copying the new 1-in-3-clauses, negating all the literals and turning them into a 2-in-3-clause. Hence, each 2-in-3-clauses evaluates to $\top$, if and only if its corresponding 1-in-3-clause does. It is not hard to verify the correctness of the reduction. Similar constructions are widely used, see, e.g., [31] or [5]. The remarkable aspect of the present construction lies in its symmetrical structure which helps to avoid additional dummy gadgets in the following reductions.

Simple Reduction. In the following, we assume that an instance of 3-SAT* with m 1-in-3-clauses $C_1, \dots, C_m$, m 2-in-3-clauses $C_{m+1}, \dots, C_{2m}$ and n variables $x_1, \dots, x_n$ is given. Note that we have $2m$ clauses with 3 literals each, and $4n$ occurring literals in total, hence $3m = 2n$. In addition to the ordering of the variables and clauses, we fix an ordering of the literals belonging to each clause, and an ordering of the occurrences of each variable by assigning an index $t \in [4]$ to each of them. In particular, for each variable x_j, $t = 1, 2$ correspond to the first and second positive and $t = 3, 4$ to the first and second negative occurrence of x_j. Furthermore, let $\kappa : [n] \times [4] \to [2m] \times [3]$ be the bijection defined as follows: $\kappa(j, t) = (i, s)$ implies that the t-th occurrence of x_j is positioned in clause C_i on position s.

We now define the restricted assignment instance. For some of the machines, we introduce *private loads* which is a synonym for jobs of the corresponding size that have to be scheduled on the respective machine because its the only eligible one. The sizes and sets of eligible machines of the introduced jobs are presented in Table 1 and the target makespan is given by $T = 322$.

- For each clause C_i, there are three *clause machines* $\texttt{CMach}_{i,s}$ with $s \in [3]$ corresponding to its three literals, as well as three *clause jobs* $\texttt{CJob}^{\circ_{s'}}_{i,s'}$ with $s' \in [3]$ and $\circ_{s'} \in \{\top, \bot\}$. We have $\circ_1 = \top$ and $\circ_3 = \bot$, as well as $\circ_2 = \bot$ if C_i is a 1-in-3 clause, and $\circ_2 = \top$ otherwise. Furthermore, each clause machine has a private load of 111.
- For each variable x_j, there are two *truth assignment machines* $\texttt{TMach}_{j,q}$ with $q \in [2]$ corresponding to the positive ($q = 1$) and negative ($q = 2$) literal of x_j, as well as 2 *truth assignment jobs* $\texttt{TJob}^{\circ}_{j}$ with $\circ \in \{\top, \bot\}$.
- For each variable x_j, there are eight *variable jobs* $\texttt{VJob}^{\circ}_{j,t}$ with $t \in [4]$ and $\circ \in \{\top, \bot\}$ corresponding to the two occurrences of the positive ($t \in \{1, 2\}$) and negative ($t \in \{3, 4\}$) literal of x_j.

Counting the different machines and jobs and adding up the job sizes yields:

▷ Claim 3. The overall size of all the jobs is exactly $|\mathcal{M}|T$.

We will show that there is a satisfying truth assignment for the 3-SAT* instance if and only if there is a schedule in which each machine receives jobs with load exactly T.

Table 2 Each set indicates one of the possible job assignments for each machine in a schedule with makespan T.

Machine	Possible Schedules
$\texttt{TMach}_{j,1}$	$\{\texttt{TJob}_j^\top, \texttt{VJob}_{j,1}^\top, \texttt{VJob}_{j,2}^\top\}$, $\{\texttt{TJob}_j^\bot, \texttt{VJob}_{j,1}^\bot, \texttt{VJob}_{j,2}^\bot\}$
$\texttt{TMach}_{j,2}$	$\{\texttt{TJob}_j^\top, \texttt{VJob}_{j,3}^\top, \texttt{VJob}_{j,4}^\top\}$, $\{\texttt{TJob}_j^\bot, \texttt{VJob}_{j,3}^\bot, \texttt{VJob}_{j,4}^\bot\}$
$\texttt{CMach}_{i,s}$ (1-in-3-clause)	$\{\texttt{VJob}_{\kappa^{-1}(i,s)}^\top, \texttt{CJob}_{i,1}^\top\}$, $\{\texttt{VJob}_{\kappa^{-1}(i,s)}^\bot, \texttt{CJob}_{i,2}^\bot\}$, $\{\texttt{VJob}_{\kappa^{-1}(i,s)}^\bot, \texttt{CJob}_{i,3}^\bot\}$
$\texttt{CMach}_{i,s}$ (2-in-3-clause)	$\{\texttt{VJob}_{\kappa^{-1}(i,s)}^\top, \texttt{CJob}_{i,1}^\top\}$, $\{\texttt{VJob}_{\kappa^{-1}(i,s)}^\top, \texttt{CJob}_{i,2}^\top\}$, $\{\texttt{VJob}_{\kappa^{-1}(i,s)}^\bot, \texttt{CJob}_{i,3}^\bot\}$

For any job $\texttt{Job}^\circ$ with $\circ \in \{\top, \bot\}$, we refer to $\circ$ as its *truth configuration* and say that $\texttt{Job}^\circ$ has $\circ$-configuration. The rationale of the reduction is as follows: Each clause machine $\texttt{CMach}_{i,s}$ should receive exactly one variable job corresponding to the literal placed in position s in the clause. The truth configuration of this variable job should correspond to the truth value the variable contributes to the clause. To ensure that the jobs $\texttt{VJob}_{j,t}^\top$ belonging to variable x_j contribute consistent truth values, the truth assignment jobs and machines are introduced. In the following, we sometimes talk about the truth assignment gadget and thus refer to these jobs and machines. Similarly, the clause machines and jobs are sometimes called the clause gadget. Note that the basic approach of using some kind of truth assignment and clause gadget for reductions in the context of restricted assignment has been used before, see, e.g., [7, 4, 5].

Next, we present a sequence of easy claims concerning the properties of a schedule for the above instance with makespan T. Due to Table 1 and Claim 3, we have:

▷ Claim 4. Each machine receives exactly 3 jobs (including private loads).

Since each digit of each occurring size is upper bounded by 2, the above claim implies that there can be no carryover when adding up job sizes of jobs scheduled on each machine. Hence the digits of the numbers involved can be considered independently, e.g., there can be at most two jobs with a 1 in the third (or second) digit of its size scheduled on any machine. This together with the given job restrictions already implies:

▷ Claim 5. Each truth assignment machine receives exactly one truth assignment and two variable jobs; and each clause machine receives exactly one clause and one variable job.

▷ Claim 6. The jobs scheduled on a truth assignment or clause machine all have the same truth configuration (excluding private loads).

▷ Claim 7. Let $j \in [n]$. The truth configuration of any job scheduled on $\texttt{TMach}_{j,1}$ is distinct from the truth configuration of any job scheduled on $\texttt{TMach}_{j,2}$.

The resulting possible schedules for each machine are summed up in Table 2, and Figure 1 depicts the resulting two possible schedules for each pair of truth assignment machines. Lastly, we have:

▷ Claim 8. For each $i \in [2m]$, the three clause machines corresponding to i receive exactly one variable job with $\top$-configuration if C_i is a 1-in-3-clause and exactly two such jobs if C_i is a 2-in-3-clause.

Using the above claims, we can easily show:

▶ **Proposition 9.** *There is a satisfying truth assignment for the given* 3-SAT* *instance if and only if there is a schedule with makespan T for the described restricted assignment instance.*

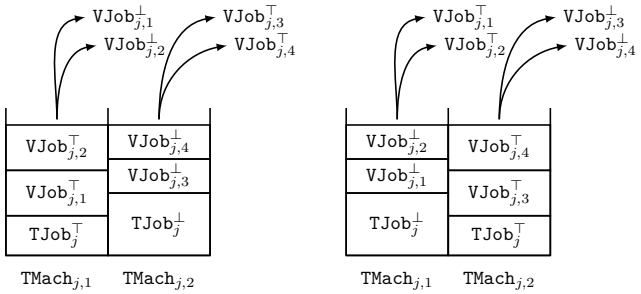

Figure 1 The truth assignment gadget: There are two possible schedules of the truth assignment machines $\texttt{TMach}_{j,1}$ and $\texttt{TMach}_{j,2}$ that already determine the schedule of the variable jobs.

Refined Reduction. When trying to adapt the above reduction to the more restricted problem of RAI, we obviously have less leeway defining the restrictions. To deal with this, we introduce additional gadgets and encode much more information into the job sizes. The idea of the reduction can be described as follows. We arrange the truth assignment gadgets on the left and the clause gadgets on the right. Consider the case that a truth assignment decision is made in the left most truth assignment gadget. Information about this decision – called signal in the following – has to be passed on to the proper clause gadgets passing multiple other truth assignment and clause gadgets on the way. This signal in the simple reduction simply corresponds to a variable job that is to be scheduled on its corresponding clause machine, and in order to prevent interaction with other gadgets, we could encode information about the corresponding variable into the size of the variable job. However, this would lead to a super constant number of job sizes. To avoid this, we introduce a new gadget called the bridge and highway gadget. Very roughly speaking, the signal is passed on to the *highway* via *gateways*; the highway passes each following truth assignment gadget using *bridges* and carries the signal to the proper clauses. Next, we give a detailed description and analysis of the refined reduction.

We adopt all the machines and jobs introduced in the simple reduction, but change the sizes and sets of eligible machines and introduce additional jobs and machines as well as private loads for *every* machine. We introduce the following jobs and machines:

- For each $j \in [n]$ and $t \in [4]$, we introduce one *gateway machine* $\texttt{GMach}_{j,t}$.
- For each $j \in [n]$, $t \in [4]$ and $j' \in \{j+1, \dots n\}$, we introduce two *bridge machines* $\texttt{BMachIn}_{j,t,j'}$ and $\texttt{BMachOut}_{j,t,j'}$. Furthermore, we introduce two *bridge jobs* $\texttt{BJob}^{\top}_{j,t,j'}$ and $\texttt{BJob}^{\perp}_{j,t,j'}$.
- For each $j \in [n]$, $t \in [4]$ and $j' \in \{j, \dots n\}$, we introduce two *highway jobs* $\texttt{HJob}^{\top}_{j,t,j'}$ and $\texttt{HJob}^{\perp}_{j,t,j'}$.

In order to define the intervals of eligible machines, we first need a total order of the machines. We partition the machines into blocks, define an internal order for each block, and then define an order of the blocks. Remember that $\kappa : [n] \times [4] \to [2m] \times [3]$ is a bijection indicating the positions of the occurrences of variables in the clauses. In particular, $\kappa(j,1) = (i,s)$ indicates that the first positive occurrence of variable x_j is in clause C_i on position s, and $\kappa(j,2)$, $\kappa(j,3)$, and $\kappa(j,4)$ indicate analogue information for the second positve, first negative, and second negative occurrence of x_j.

- For each $j \in [n]$, we have a truth assignment block $\mathcal{T}_j$ containing the truth assignment machines $\texttt{TMach}_{j,1}$ and $\texttt{TMach}_{j,2}$ in this order.
- For each $i \in [2m]$, we have a clause block $\mathcal{C}_i$ containing the clause machines $\texttt{CMach}_{i,s}$ for each $s \in [3]$ and ordered increasingly by s.

- For each $j \in [n]$, we have a successor block $\mathcal{S}_j$ containing the gateway machines $\texttt{GMach}_{j,t}$ for each $t \in [4]$ and the bridge machines $\texttt{BMachOut}_{j',t,j}$ for each $t \in [4]$ and $j' < j$. For each machine, we define an index, namely $\kappa(j,t)$ for $\texttt{GMach}_{j,t}$ and $\kappa(j',t)$ for $\texttt{BMachOut}_{j',t,j}$, and order the machines by the *decreasing* lexicographical ordering of their indices. For example, if $\texttt{BMachOut}_{j_1,t_1,j}, \texttt{BMachOut}_{j_2,t_2,j}, \texttt{GMach}_{j,t_3} \in \mathcal{S}_j$ and $\kappa(j_1,t_1) = (1,2)$, $\kappa(j_2,t_2) = (1,1)$ and $\kappa(j,t_3) = (2,3)$, then $\texttt{GMach}_{j,t_3}$ precedes $\texttt{BMachOut}_{j_1,t_1,j}$ which in turn precedes $\texttt{BMachOut}_{j_2,t_2,j}$.
- For each $j \in [n]$ with $j > 1$, we have a predecessor block $\mathcal{P}_j$ containing the bridge machines $\texttt{BMachIn}_{j',t,j}$ for each $t \in [4]$ and $j' < j$. Machine $\texttt{BMachIn}_{j',t,j}$ has index $\kappa(j',t)$ and the machines are ordered by the *increasing* lexicographical ordering of their indices.

The blocks are ordered as follows:

$$(\mathcal{T}_1, \mathcal{S}_1, \mathcal{P}_2, \mathcal{T}_2, \mathcal{S}_2, \ldots, \mathcal{P}_n, \mathcal{T}_n, \mathcal{S}_n, \mathcal{C}_1, \ldots, \mathcal{C}_{2m})$$

The sets of eligible machines are specified in Table 3, and in Table 4 the job sizes as well as the target makespan T are given[3]. Figure 2 gives some intuition on the overall structure. Due to counting and basic arithmetic, we have:

▷ Claim 10. The overall size of the jobs is exactly $|\mathcal{M}|T$.

Like for the simple reduction, we prove a sequence of easy claims concerning the properties of a schedule for the constructed instance with makespan T. First note:

▷ Claim 11. Each machine receives exactly 4 jobs if it is a truth assignment machine and exactly 3 jobs otherwise (including private loads).

Since each machine receives at most 4 jobs and each digit in the job sizes is bounded by 2, we may consider each digit of the involved numbers independently, e.g., if two jobs and the makespan have a 1 at the ℓ-th digit, we already know that these jobs cannot be scheduled on the same machine. This already implies a series of claims:

▷ Claim 12. The jobs $\texttt{TJob}_j^{\top}$ and $\texttt{TJob}_j^{\bot}$ can exclusively be scheduled on $\texttt{TMach}_{j,1}$ and $\texttt{TMach}_{j,2}$, for each $j \in [n]$, and each of the two machines receives exactly one of the two jobs.

▷ Claim 13. The jobs $\texttt{VJob}_{j,t}^{\top}$ and $\texttt{VJob}_{j,t}^{\bot}$ can exclusively be scheduled on $\texttt{TMach}_{j,\lceil t/2 \rceil}$ and $\texttt{GMach}_{j,t}$, for each $j \in [n]$ and $t \in [4]$, and each of the two machines receives exactly one of the two jobs.

▷ Claim 14. Bridge jobs can exclusively be scheduled on bridge machines and each bridge machine receives exactly one bridge job.

▷ Claim 15. Highway jobs can exclusively be scheduled on bridge, gateway and clause machines and each such machine receives exactly one highway job.

▷ Claim 16. Each clause machine $\texttt{CMach}_{i,s}$ receives exactly one of the corresponding clause jobs $\texttt{CJob}_{i,s'}^{\circ_{s'}}$ with $s' \in [3]$.

At this point, we already know that variable (and truth assignment) jobs can exclusively be scheduled on the first or last machine of their respective interval of eligible machines. The next step is to show that the same holds for highway and bridge jobs. To do so, the ordering of the bridge and highway machines is of critical importance.

[3] Note that we have prioritized comprehensibility over small sizes. For instance, it is not hard to see that the columns in Table 4 corresponding to the highway and clause jobs could be deleted and the reduction would still work.

Table 3 The sets of eligible machines for each job or job type, defined by the first and last eligible machine in the ordering. Note that in case of the highway jobs all four combinations of first and last machine are possible.

Job	First machine	Last machine
Clause job $\texttt{CJob}^{\circ s}_{i,s}$	$\texttt{CMach}_{i,1}$	$\texttt{CMach}_{i,3}$
Truth assignment job $\texttt{TJob}^{\circ}_{j}$	$\texttt{TMach}_{j,1}$	$\texttt{TMach}_{j,2}$
Variable job $\texttt{VJob}^{\circ}_{j,t}$	$\texttt{TMach}_{j,\lceil t/2 \rceil}$	$\texttt{GMach}_{j,t}$
Bridge job $\texttt{BJob}^{\circ}_{j,t,j'}$	$\texttt{BMachIn}_{j,t,j'}$	$\texttt{BMachOut}_{j,t,j'}$
Highway job $\texttt{HJob}^{\circ}_{j,t,j'}$	$\texttt{BMachOut}_{j,t,j'}$, if $j' > j$, $\texttt{GMach}_{j,t}$, if $j' = j$	$\texttt{BMachIn}_{j,t,j'+1}$ if $j' < n$, $\texttt{CMach}_{\kappa(j,t)}$, if $j' = n$

Table 4 Table of job and machine types with job sizes and private loads and the makespan. The second column states the number of jobs and machines of the respective types. Each horizontal sequence of numbers following the second column indicates the size of the respective job or private load. Each of the corresponding columns serves a function in the reduction: the first bounds the number of jobs on each machines; the following eight implement restrictions for the bridge, highway, clause, truth assignment and variable jobs; and the last encodes truth values.

	#		B	H	C	T	V	V	V	V	
$\texttt{CJob}^{\top}_{i,s}$	$3m = 2n$	1	0	0	1	0	0	0	0	0	0
$\texttt{CJob}^{\bot}_{i,s}$	$3m = 2n$	1	0	0	1	0	0	0	0	0	1
$\texttt{TJob}^{\top}_{j}$	n	1	0	0	0	1	0	0	0	0	2
$\texttt{TJob}^{\bot}_{j}$	n	1	0	0	0	1	0	0	0	0	0
$\texttt{VJob}^{\top}_{j,1}$	n	1	0	0	0	0	1	0	0	0	0
$\texttt{VJob}^{\top}_{j,2}$	n	1	0	0	0	0	0	1	0	0	0
$\texttt{VJob}^{\top}_{j,3}$	n	1	0	0	0	0	0	0	1	0	0
$\texttt{VJob}^{\top}_{j,4}$	n	1	0	0	0	0	0	0	0	1	0
$\texttt{VJob}^{\bot}_{j,1}$	n	1	0	0	0	0	1	0	0	0	1
$\texttt{VJob}^{\bot}_{j,2}$	n	1	0	0	0	0	0	1	0	0	1
$\texttt{VJob}^{\bot}_{j,3}$	n	1	0	0	0	0	0	0	1	0	1
$\texttt{VJob}^{\bot}_{j,4}$	n	1	0	0	0	0	0	0	0	1	1
$\texttt{BJob}^{\top}_{j,t,j'}$	$2n(n-1)$	1	1	0	0	0	0	0	0	0	0
$\texttt{BJob}^{\bot}_{j,t,j'}$	$2n(n-1)$	1	1	0	0	0	0	0	0	0	1
$\texttt{HJob}^{\top}_{j,t,j'}$	$2n(n+1)$	1	0	1	0	0	0	0	0	0	1
$\texttt{HJob}^{\bot}_{j,t,j'}$	$2n(n+1)$	1	0	1	0	0	0	0	0	0	0
$\texttt{CMach}_{i,s}$	$6m = 4n$	1	1	0	0	1	1	1	1	1	1
$\texttt{TMach}_{j,1}$	n	0	1	1	1	0	0	0	1	1	0
$\texttt{TMach}_{j,2}$	n	0	1	1	1	0	1	1	0	0	0
$\texttt{GMach}_{j,1}$	n	1	1	0	1	1	0	1	1	1	1
$\texttt{GMach}_{j,2}$	n	1	1	0	1	1	1	0	1	1	1
$\texttt{GMach}_{j,3}$	n	1	1	0	1	1	1	1	0	1	1
$\texttt{GMach}_{j,4}$	n	1	1	0	1	1	1	1	1	0	1
$\texttt{BMachIn}_{j,t,j'}$	$2n(n-1)$	1	0	0	1	1	1	1	1	1	1
$\texttt{BMachOut}_{j,t,j'}$	$2n(n-1)$	1	0	0	1	1	1	1	1	1	1
Makespan T		3	1	1	1	1	1	1	1	1	2

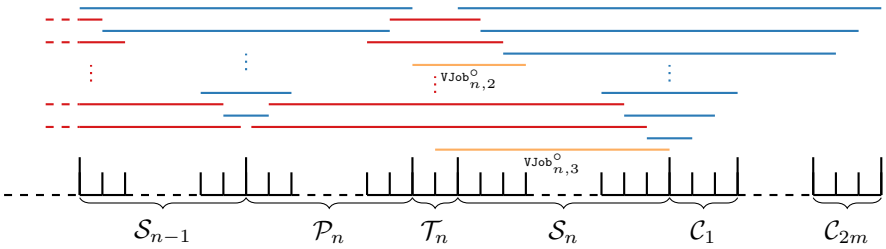

Figure 2 The bridge and highway gadget. The intervals of eligible machines of highway, bridge and variable jobs are depicted in blue, red and orange, respectively. In this example, variable x_n occurs for the second time in its positive form in the last clause at the first position, and for the first time in its negative form in the first clause at the first position.

▷ Claim 17. The jobs $\texttt{BJob}^{\top}_{j,t,j'}$ and $\texttt{BJob}^{\bot}_{j,t,j'}$ can exclusively be scheduled on $\texttt{BMachIn}_{j,t,j'}$ and $\texttt{BMachOut}_{j,t,j'}$, for each $j \in [n]$, $j' \in \{j+1,\dots,n\}$ and $t \in [4]$, and each of the two machines receives exactly one of the two jobs.

Proof. The claim can be proved with a simple inductive argument: Let $j' \in \{2,\dots,n\}$ and, furthermore, $(j_\ell, t_\ell) \in [j'-1] \times [4]$ denote the ℓ-th element from $[j'-1] \times [4]$ when ordering the pairs $(j,t) \in [j'-1] \times [4]$ by the increasing lexicographical ordering of the pairs $\kappa(j,t)$. Considering the ordering of the machines and the job restrictions, $\texttt{BJob}^{\top}_{j_1,t_1,j'}$ and $\texttt{BJob}^{\bot}_{j_1,t_1,j'}$ are the only bridge jobs that can be scheduled on $\texttt{BMachIn}_{j_1,t_1,j'}$ and $\texttt{BMachOut}_{j_1,t_1,j'}$ (see Figure 2). Hence, the claim has to hold for (j_1,t_1). But then again $\texttt{BJob}^{\top}_{j_2,t_2,j'}$ and $\texttt{BJob}^{\bot}_{j_2,t_2,j'}$ are the only remaining bridge jobs that can be scheduled on $\texttt{BMachIn}_{j_2,t_2,j'}$ and $\texttt{BMachOut}_{j_2,t_2,j'}$, and so on. ◁

▷ Claim 18. The jobs $\texttt{HJob}^{\top}_{j,t,j'}$ and $\texttt{HJob}^{\bot}_{j,t,j'}$ can exclusively be scheduled on machine $\texttt{X}$ and $\texttt{Y}$, for each $j \in [n]$, $j' \geq j$, and $t \in [4]$; where $\texttt{X} = \texttt{BMachOut}_{j,t,j'}$ if $j' > j$, and $\texttt{X} = \texttt{GMach}_{j,t}$ otherwise, and $\texttt{Y} = \texttt{BMachIn}_{j,t,j'+1}$ if $j' < n$, and $\texttt{Y} = \texttt{CMach}_{\kappa(j,t)}$ otherwise. Furthermore, each of the two machines receives exactly one of the two jobs.

Proof. We can use the same argument (with reversed orderings) as we did in the last claim. It is only slightly more complicated, because more machine types are involved. ◁

Summing up, each job except for clause jobs may only be scheduled on the first or last machine of their interval of eligible machines, and each of these machines receives either the respective job in $\top$- or $\bot$-configuration. Considering this distribution of the jobs and the last digit of the size vectors, we get the following two claims:

▷ Claim 19. For any machine, the jobs assigned to this machine all have the same truth configuration (excluding private loads).

▷ Claim 20. For each $i \in [2m]$, the three clause machines corresponding to i receive exactly one highway job with $\top$-configuration, if C_i is a 1-in-3-clause, and exactly two such jobs, if C_i is a 2-in-3-clause.

The former property together with the possible job distribution determined so far implies that there are only few possible schedules for each machine. We summarize these schedules in Table 5. Furthermore, we can infer that the truth assignment gadget works essentially the same as before (see Figure 1):

▷ Claim 21. Let $j \in [n]$. The truth configuration of any job scheduled on $\texttt{TMach}_{j,1}$ is distinct from the truth configuration of any job scheduled on $\texttt{TMach}_{j,2}$.

Table 5 For each machine there are only few possible jobs that may be assigned to it in a schedule with makespan T. Each set corresponds to one of the possible schedules.

Machine	Possible Schedule
$\texttt{TMach}_{j,1}$	$\{\texttt{TJob}_j^{\top}, \texttt{VJob}_{j,1}^{\top}, \texttt{VJob}_{j,2}^{\top}\}$, $\{\texttt{TJob}_j^{\bot}, \texttt{VJob}_{j,1}^{\bot}, \texttt{VJob}_{j,2}^{\bot}\}$
$\texttt{TMach}_{j,2}$	$\{\texttt{TJob}_j^{\top}, \texttt{VJob}_{j,3}^{\top}, \texttt{VJob}_{j,4}^{\top}\}$, $\{\texttt{TJob}_j^{\bot}, \texttt{VJob}_{j,3}^{\bot}, \texttt{VJob}_{j,4}^{\bot}\}$
$\texttt{GMach}_{j,t}$	$\{\texttt{VJob}_{j,t}^{\top}, \texttt{HJob}_{j,t,j}^{\top}\}$, $\{\texttt{VJob}_{j,t}^{\bot}, \texttt{HJob}_{j,t,j}^{\bot}\}$
$\texttt{BMachIn}_{j,t,j'}$	$\{\texttt{BJob}_{j,t,j'}^{\top}, \texttt{HJob}_{j,t,j'-1}^{\top}\}$, $\{\texttt{BJob}_{j,t,j'}^{\bot}, \texttt{HJob}_{j,t,j'-1}^{\bot}\}$
$\texttt{BMachOut}_{j,t,j'}$	$\{\texttt{BJob}_{j,t,j'}^{\top}, \texttt{HJob}_{j,t,j'}^{\top}\}$, $\{\texttt{BJob}_{j,t,j'}^{\bot}, \texttt{HJob}_{j,t,j'}^{\bot}\}$
$\texttt{CMach}_{i,s}$ (1-in-3)	$\{\texttt{CJob}_{i,1}^{\top}, \texttt{HJob}_{\kappa^{-1}(i,s),n}^{\top}\}$, $\{\texttt{CJob}_{i,2}^{\bot}, \texttt{HJob}_{\kappa^{-1}(i,s),n}^{\bot}\}$,$\{\texttt{CJob}_{i,3}^{\bot}, \texttt{HJob}_{\kappa^{-1}(i,s),n}^{\bot}\}$
$\texttt{CMach}_{i,s}$ (2-in-3)	$\{\texttt{CJob}_{i,1}^{\top}, \texttt{HJob}_{\kappa^{-1}(i,s),n}^{\top}\}$, $\{\texttt{CJob}_{i,2}^{\top}, \texttt{HJob}_{\kappa^{-1}(i,s),n}^{\top}\}$,$\{\texttt{CJob}_{i,3}^{\bot}, \texttt{HJob}_{\kappa^{-1}(i,s),n}^{\bot}\}$

Lastly, we can show that the bridge and highway gadget works as well:

▷ Claim 22. Let $j \in [n]$ and $t \in [4]$. The variable job scheduled on $\texttt{TMach}_{j,\lceil t/2 \rceil}$ and the highway job scheduled on $\texttt{CMach}_{\kappa(j,t)}$ have the same truth configuration.

Proof. Note that the truth configuration of the variable job scheduled on $\texttt{GMach}_{j,t}$ compared with the one of the variable job scheduled on $\texttt{TMach}_{j,\lceil t/2 \rceil}$ is reversed. Hence, the highway job scheduled on $\texttt{GMach}_{j,t}$ also has the reversed truth-configuration while the highway job that is passed on again has the original truth-configuration. This argument can be repeated with the bridge and highway jobs in the following, yielding the asserted claim. ◁

Using the above claims, we can conclude the proof of Theorem 1 via the following Lemma:

▶ **Lemma 23.** *There is a satisfying truth assignment for the given* 3-SAT* *instance, if and only if there is a schedule with makespan T for the constructed* RAI *instance.*

Proof. First, we consider the case that a schedule with makespan T for the constructed RAI instance is given. For each variable x_j and occurrence $t \in [4]$, let $\texttt{HJob}_{j,t,n}^{\circ_{j,t}}$ be the highway job scheduled on $\texttt{CMach}_{\kappa(j,t)}$ (see Table 5). We choose the truth value of x_j to be $\circ_{j,1}$. Considering the distribution of jobs on the truth assignment machines (see Table 5), as well as Claim 21 and 22, we know that for each variable x_j and occurrence $t \in [4]$, the truth configuration $\circ_{j,t}$ corresponds exactly to the truth value x_j contributes to the clause given by $\kappa(j,t)$. Furthermore, we know that for each clause C_i, there are exactly three variable jobs scheduled on the corresponding clause machines, and exactly one or two of these has $\top$-configuration, if C_i is a 1-in-3-clause or 2-in-3-clause, respectively (Claim 20). Hence, C_i is satisfied.

Now, let there be a satisfying truth assignment, and $\triangleleft_j$ be the corresponding truth value of variable x_j and $\triangleright_j$ its negation. We set $\triangle_{j,t} = \triangleleft_j$ for $t \in \{1,2\}$ and $\triangle_{j,t} = \triangleright_j$ for $t \in \{3,4\}$ and assign $\texttt{HJob}_{j,t,n}^{\triangle_{j,t}}$ to $\texttt{CMach}_{\kappa(j,t)}$. Let $\triangledown_{jt}$ be the negation of $\triangle_{jt}$. All the other jobs are assigned as indicated by the claims and Table 5 in particular: Each machine receives its private load; $\texttt{CMach}_{\kappa(j,t)}$ additionally receives one of the eligible remaining clause jobs with $\triangle_{j,t}$-configuration (this can be done because the truth assignment is satisfying); $\texttt{BMachOut}_{j,t,j'}$ receives $\texttt{HJob}_{j,t,j'}^{\triangledown_{j,t}}$ and $\texttt{BJob}_{j,t,j'}^{\triangledown_{j,t}}$; $\texttt{BMachIn}_{j,t,j'}$ receives $\texttt{HJob}_{j,t,j'-1}^{\triangle_{j,t}}$ and $\texttt{BJob}_{j,t,j'}^{\triangle_{j,t}}$; $\texttt{GMach}_{j,t}$ receives $\texttt{HJob}_{j,t,j}^{\triangledown_{j,t}}$ and $\texttt{VJob}_{j,t}^{\triangledown_{j,t}}$; $\texttt{TMach}_{j,1}$ receives $\texttt{VJob}_{j,1}^{\triangle_{j,1}}$, $\texttt{VJob}_{j,2}^{\triangle_{j,2}}$ and $\texttt{TJob}_{j}^{\triangle_{j,1}}$; and $\texttt{TMach}_{j,2}$ receives $\texttt{VJob}_{j,3}^{\triangle_{j,3}}$, $\texttt{VJob}_{j,4}^{\triangle_{j,4}}$ as well as $\texttt{TJob}_{j}^{\triangle_{j,3}}$. It is easy to verify, that all jobs are assigned and each machine has a load of T. ◀

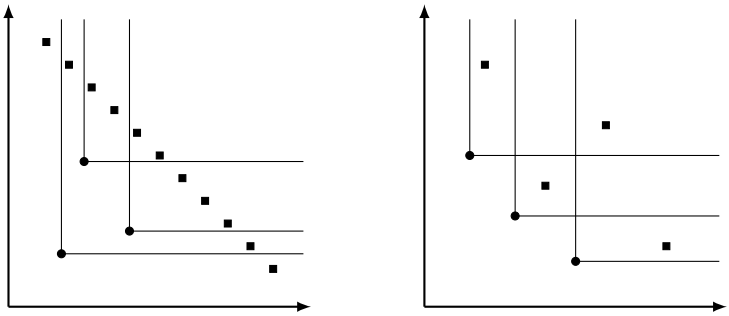

Figure 3 The left picture visualizes that each RAI instance can be seen as a RAR(2) instance and the right one depicts an RAR(2) instance that is not a RAI instance. In both pictures, each dimension corresponds to a resource, the squares mark the capacities of machines and the circles the demands of jobs. If the capacity of a machine is at least as big as the demand of a job in both dimension, the job is eligible on the machine.

3 Resource Restrictions

In this section, we briefly discuss scheduling with resource restrictions and provide the proof of Theorem 2 in particular. First note that RAI is properly placed between RAR(1) and RAR(2), that is, with a slight abuse of notation, RAR(1) $\subset$ RAI $\subset$ RAR(2). The ideas needed to see RAI $\subset$ RAR(2) are given in Figure 3.

The result in Theorem 2 concerning 4 resources is proven by showing that the restrictions in a reduction due to Ebenlendr et al. [7] can be modeled using 4 resources. Concerning the result for 2 resources, we first discuss the corresponding result for 3 resources. The reduction is based on the classical result by Lenstra et al. [19] and very similar to a reduction by Bhaskara et al. [1] for rank four unrelated scheduling. However, there is a problem with the choice of processing times in the latter reduction (see [22]), and the present result can be used to fix it. Combining the ideas in that reduction with a result by Chen et al. [6] yields the result for 2 resources.

Four Resources. In the classical 3-SAT problem, a conjunction of m clauses is given and each clause is a disjunction of at most three literals of variables $x_1, \ldots, x_n$. In the result due to Ebenlendr et al. [7], the modified 3-SAT problem, where each variable occurs exactly three and each literal at most two times in the formula, is reduced to the graph balancing problem, that is, restricted assignment with the additional property that each job is eligible on at most two machines. To show that the modified 3-SAT problem is NP-hard, we can use techniques already applied in Section 2: We may replace the d_j occurrences of variable x_j with new variables $z_{j1}, \ldots, z_{jd_j}$ and add new clauses $(z_{j1} \wedge \neg z_{j2}), \ldots (z_{jd_{j-1}} \wedge \neg z_{jd_j}), (z_{jd_j} \wedge \neg z_{j1})$.

▶ **Theorem 24** ([7]). *There is no polynomial time approximation algorithm with rate smaller than* 1.5 *for the graph balancing problem unless P=NP.*

Proof. Given an instance of modified 3-SAT, we introduce clause machines v_i corresponding to the clauses C_i, and literal machines $u_{j,1}$ and $u_{j,0}$ corresponding to the literals x_j and $\neg x_j$. Furthermore, we introduce truth assignment jobs e_j for each variable x_j with size 2 and eligible on $u_{j,1}$ and $u_{j,0}$; and clause jobs $f_{i,j,\alpha}$ for each clause C_i and literal y_j occurring in C_i with $\alpha = 1$ if $y_j = x_j$ and $\alpha = 0$ if $y_j = \neg x_j$. The job $f_{i,j,\alpha}$ has size 1 and is eligible on v_i and $u_{j,\alpha}$. Lastly, we introduce a dummy job d_i for each clause C_i with less then three literals. Its size is 1 if C_i contains two literals, and 2 if C_i contains only one literal.

In a schedule with makespan 2, there is at least one clause job $f_{i,j,\alpha}$ for each v_i that is scheduled on $u_{j,\alpha}$ and not on v_i. Hence, the job e_j has to be scheduled on $u_{j,|\alpha-1|}$. Now, it is easy to see that there is a schedule with makespan 2, if and only if there is a satisfying assignment. The construction works as follows: Given a schedule with makespan 2, we set variable x_j to $\top$ if e_j is scheduled on $u_{j,0}$, and to $\bot$ otherwise. Moreover, given a satisfying truth assignment we assign the truth assignment jobs correspondingly, and the machines $u_{j,\alpha}$ that did not receive a truth assignment job receives all eligible clause jobs (at most two). ◀

We reproduce the restrictions in the above reduction using four resources and get:

▶ **Corollary 25.** *There is no polynomial time approximation algorithm for* RAR(4) *with rate smaller than* 1.5 *unless P=NP.*

Proof. We set $\mathcal{R} = [4]$. The clause machine v_i has a resource capacity vector of $(2n + 1, 2n + 1, i, (m + 1) - i)$, and the literal machine $u_{j,\alpha}$ has capacity vectors $(2j - \alpha, (2n + 1) - (2j - \alpha), m + 1, m + 1)$. Furthermore, the truth assignment job e_j has a resource demand vector of $(2j - 1, (2n + 1) - 2j, m + 1, m + 1)$; the clause job $f_{i,j,\alpha}$ has a demand vector of $(2j - \alpha, (2n + 1) - (2j - \alpha), i, (m + 1) - i)$; and the dummy job d_i has a demand vector of $(2n + 1, 2n + 1, i, (m + 1) - i)$. It is easy to verify that the resulting sets of eligible machines are the same as described in Theorem 24. ◀

Three Resources. In the 3-DM problem, the input consists of three disjoint sets A, B and C with $|A| = |B| = |C| = n \in \mathbb{N}$, as well as a set of triplets $E \subseteq \big\{\{a, b, c\} \,\big|\, a \in A, b \in B, c \in C\big\}$. The goal is to decide whether there is a subset $F \subseteq E$ that perfectly covers A, B and C, that is, for each $x \in A \cup B \cup C$ there is exactly one triplet $e \in F$ with $x \in e$. The set F is called a 3D-matching. We assume that the elements of A, B and C are indexed, that is, $A = \{a_1, a_2, \ldots, a_n\}$, $B = \{b_1, b_2, \ldots, b_n\}$ and $C = \{c_1, c_2, \ldots, c_n\}$. Furthermore, we assume that for each $x \in A \cup B \cup C$ there is at least one $e \in E$ with $x \in E$.

We present a reduction from 3-DM to RAR(3). Given an instance (A, B, C, E) of 3-DM, let $E(x) = \{e \in E \,|\, x \in e\}$ for each $x \in A \cup B \cup C$. Furthermore, we set $\alpha_A = 12$, $\alpha_B = 13$, $\alpha_C = 22$, $\beta_A = 14$, $\beta_B = 15$ and $\beta_C = 18$. Let $\mathcal{R} = \{A, B, C\}$ and $\mathcal{M} = E$. For each machine e, we define the resource capacities as follows. Let $X \in \{A, B, C\}$ and $x_i \in X \cap e$ be the element of x with index i. We set $c_X(e) = i$. Furthermore, for each element $x_i \in X$ with index i in $X \in \{A, B, C\}$, we introduce one element job with size α_X and $|E(x)| - 1$ dummy jobs with size β_X. The resource demand for each of these jobs is given by $d(i)$ with $d_X(i) = i$ and $d_Y(i) = 0$ for $Y \in \{A, B, C\} \setminus \{X\}$.

▷ Claim 26. We have $\alpha_A + \alpha_B + \alpha_C = 47 = \beta_A + \beta_B + \beta_C$; any four numbers taken from $\Gamma = \{\alpha_A, \alpha_B, \alpha_C, \beta_A, \beta_B, \beta_C\} = \{12, 13, 22, 14, 15, 18\}$ sum up to a value bigger than 47; any selection of less than 3 numbers sums up to a value smaller than 47; and for any three numbers $\gamma_1, \gamma_2, \gamma_3 \in \Gamma$ with $\gamma_1 \leq \gamma_2 \leq \gamma_3$ and $\gamma_1 + \gamma_2 + \gamma_3 = 47$, we have either $(\gamma_1, \gamma_2, \gamma_3) = (\alpha_A, \alpha_B, \alpha_C)$ or $(\gamma_1, \gamma_2, \gamma_3) = (\beta_A, \beta_B, \beta_C)$.

Proof. The first three assertions are obvious, and the fourth holds due to a simple case analysis:

- If $\gamma_1 > 15$, we have $\gamma_1 \geq 18$, and hence $47 = \gamma_1 + \gamma_2 + \gamma_3 \geq 3 \cdot \gamma_1 = 54$: a contradiction.
- Note that $\gamma_3 \geq (\gamma_2 + \gamma_3)/2 = (47 - \gamma_1)/2$. Hence, $\gamma_1 \leq 15$ implies $\gamma_3 \geq 16$ and therefore $\gamma_3 \in \{18, 22\}$.
- If we have $\gamma_3 = 22 = \alpha_C$, then $\gamma_1 \leq (\gamma_1+\gamma_2)/2 = (47-\gamma_3)/2 = 12.5$. Hence, $\gamma_1 = 12 = \alpha_A$ and $\gamma_2 = 13 = \alpha_B$.

- If we have $\gamma_3 = 18 = \beta_C$, than $\gamma_2 \geq (\gamma_1+\gamma_2)/2 = (47-\gamma_3)/2 = 14.5$. Hence, $\gamma_2 \in \{15, 18\}$. If $\gamma_2 = 15 = \beta_B$, then $\gamma_1 = 14 = \beta_A$, and if $\gamma_2 = 18$, then $\gamma_1 = 11 \notin \Gamma$.

This concludes the proof of the claim. ◁

By brute force, it can be verified that 47 is the smallest value such that suitable numbers α_A, α_B, α_C, β_A, β_B and β_C exist and the above claim holds.

▷ Claim 27. The summed up size of all the element and dummy jobs is $47|\mathcal{M}|$.

Proof. We have exactly n element jobs with size α_A, α_B and α_C, respectively, yielding an overall load of $47n$. The dummy jobs have an overall load of:

$$\begin{aligned}&\beta_A \sum_{a\in A}(|E(a)|-1) + \beta_B \sum_{b\in B}(|E(b)|-1) + \beta_C \sum_{c\in C}(|E(b)|-1)\\ =&(\beta_A+\beta_B+\beta_C)(|E|-n) = 47(|\mathcal{M}|-n)\end{aligned}$$

In this equation, we used the simple fact that $\{E(x) \mid x \in X\}$ is a partition of E for each $X \in \{A, B, C\}$, and hence $|E| = \sum_{x\in X} |E(x)|$. ◁

These two claims imply:

▷ Claim 28. In any schedule for the constructed instance with makespan 47, each machine receives exactly three jobs with sizes $\gamma_1, \gamma_2, \gamma_3$ such that $(\gamma_1, \gamma_2, \gamma_3) = (\alpha_A, \alpha_B, \alpha_C)$ or $(\gamma_1, \gamma_2, \gamma_3) = (\beta_A, \beta_B, \beta_C)$.

Using these claims, we can show:

▶ **Proposition 29.** *There is a perfect matching for the given* 3-DM *instance, if and only if there is a schedule with makespan* 47 *for the constructed* RAR(3) *instance.*

Proof. Let F be a perfect matching for the 3-DM instance. For each $x \in A \cup B \cup C$ we assign the corresponding element job to the machine e with $x \in e$ and $e \in F$. Furthermore, the dummy jobs corresponding to $x \in X$ with $X \in \{A, B, C\}$, are distributed to the machines e with $x \in e$ and $e \notin F$ such that each machine receives exactly one job in this step. Hence, each machine $e \in E$ receives exactly three eligible jobs either with sizes α_A, α_B and α_C (if $e \in F$) or β_A, β_B and β_C (otherwise).

Next, we assume that there is a schedule with makespan 47 for the scheduling instance. For each $X \in \{A, B, C\}$, there are exactly $|\mathcal{M}|$ many jobs with size α_X or β_X, and due to the above claims, we know that each machine receives exactly one of these jobs. For each $j \in [n]$, let $x_j \in X$ be the element with index j in $X \in \{A, B, C\}$. The machines $\bigcup_{j=i}^{n} E(x_j)$ are the only machines that may process jobs corresponding to $x_i, \ldots, x_n$ for each $i \in [n]$ and we have exactly $\sum_{j=i}^{n} |E(x_j)|$ many such jobs. Hence, the machines from $E(x_i)$ receive exactly the jobs corresponding to x_i. Now, considering this and Claim 28, we get a perfect matching by selecting the machines that process three element jobs. ◀

Two Resources. We are able to refine the result for three resources to work for two resources as well by using another variant of 3-DM as the starting point of the reduction. The problem 3-DM* was introduced by Chen et al. [6] to get an improved lower bound for the approximation ratio of rank four unrelated scheduling.

In this problem, a set of six disjoint sets $\mathcal{E} = \{A, A', B, B', C, C'\}$ is given. For each $X \in \mathcal{E}$, we have $|X| = 3n$ for some $n \in \mathbb{N}$ and the sets are indexed by $[3n]$, e.g., $A = \{a_1, a_2, \ldots, a_{3n}\}$. Furthermore, there are two sets of triplets $E_1 \subseteq \big\{\{a_i, b_j, c_j\}, \{a'_i, b_j, c_j\} \,\big|\, i \in [3n], j \in [3n]\big\}$

Table 6 The resource demands and capacities for the different job (types) and machines.

Jobs	Resources	Machines	Resources
a_i	$(2i, 0)$	$\{a_i, b_j, c_j\}$	$(2i, 3n + j)$
a'_i	$(2i - 1, 0)$	$\{a'_i, b_j, c_j\}$	$(2i - 1, 3n + j)$
b_j	$(0, 3n + j)$	$\{a_i, b'_i, c'_i\}$	$(2i, i)$
c_j	$(0, 3n + j)$	$\{a'_i, b'_i, c'_{\zeta(i)}\}$	$(2i - 1, \zeta(i))$
b'_i	$(2i - 1, 0)$		
c'_i	$(0, i)$		

and $E_2 = \left\{\{a_i, b'_i, c'_i\}, \{a'_i, b'_i, c'_{\zeta(i)}\} \,\middle|\, i \in [3n]\right\}$ with $\zeta(3k + 1) = 3k + 2$, $\zeta(3k + 2) = 3k + 3$ and $\zeta(3k + 3) = 3k + 1$ for each $k \in \{0, \dots, n - 1\}$. Note that the second set of triplets is determined by the element sets in the input. Similar to the classical 3-DM problem, the goal is to decide whether there is a subset $F \subseteq E_1 \cup E_2$ that perfectly covers the element set, that is, for each $x \in \bigcup_{X \in \mathcal{E}} X$ there is exactly one triplet $e \in F$ with $x \in e$. We assume that for each $x \in \bigcup_{X \in \mathcal{E}} X$ there is at least one $e \in E$ with $x \in E$ (otherwise the problem is trivial).

Let $\alpha_A = \alpha_{A'} = 12$, $\alpha_B = \alpha_{B'} = 13$, $\alpha_C = \alpha_{C'} = 22$, $\beta_A = \beta_{A'} = 14$, $\beta_B = \beta_{B'} = 15$ and $\beta_C = \beta_{C'} = 18$. We set $\mathcal{M} = E_1 \cup E_2$ and $\mathcal{R} = [2]$. The corresponding resource capacity vectors are presented in Table 6. Furthermore, for each element $x \in X$ in $X \in \mathcal{E}$, we introduce one element job with size α_X and $|E(x)| - 1$ dummy jobs with size β_X. The vector of resource demands for each such job is given in Table 6. Note that Claim 26-28 hold for this reduction as well and with the same reasoning. A simple case analysis yields:

▷ Claim 30. For each $x \in \bigcup_{X \in \mathcal{E}} X$, a (dummy or element) job corresponding to x is eligible on each machine e with $x \in e$.

Using these claims, we can conclude the proof of Theorem 2:

▶ **Lemma 31.** *There is a perfect matching for the given* 3-DM* *instance, if and only if there is a schedule with makespan* 47 *for the constructed* RAR(2) *instance.*

Proof. Let F be a perfect matching for the 3-DM* instance. For each $x \in \bigcup_{X \in \mathcal{E}} X$, we assign the corresponding element job to the machine e with $x \in e$ and $e \in F$. Furthermore, the dummy jobs corresponding to $x \in X$ with $X \in \mathcal{E}$, are distributed to the machines e with $x \in e$ and $e \notin F$ such that each such machine receives exactly one job. Hence, each machine $e \in E$ receives exactly three eligible jobs either with sizes α_A, α_B and α_C or β_A, β_B and β_C.

Next, we assume that there is a schedule with makespan 47 for the scheduling instance. There are exactly $|\mathcal{M}|$ many jobs with size $\alpha_A = \alpha_{A'}$ or $\beta_A = \beta_{A'}$ corresponding to elements of $A \cup A'$, and due to Claim 28 we know that each machine receives exactly one of these jobs. The machines corresponding to triplets from $E(a_{3n})$ are the only ones that can process the $|E(a_{3n})|$ jobs corresponding to a_{3n}, and hence each of these machines receives exactly one of these jobs. Now, the machines corresponding to triplets from $E(a'_{3n})$ are the only *remaining* ones that can process the $|E(a'_{3n})|$ jobs corresponding to a'_{3n}. Iterating this argument, we get that each machine e receives exactly one job corresponding to some $x \in A \cup A'$ with $x \in e$. Note that the above argument was based on the first resource value. Considering the second resource value yields the same result for each $x \in C \cup C'$. For the elements $x \in B \cup B'$ both resource values have to be considered, namely the second for $b \in B$ and the first for $b' \in B'$, but the argument stays the same. Summing up, each machine $e = \{x, y, z\}$ receives exactly three jobs corresponding to x, y and z. Now, considering this and Claim 28, we get a perfect matching by selecting the triplets e that processes three element jobs. ◀

4 Conclusion

In this paper we provided hardness of approximation results for scheduling with interval and resource restrictions. We list some possible future research directions:

From the perspective of complexity, tighter hardness results seem plausible. In particular, we have the same inapproximability results for RAR(2) and RAR(3) and it would be interesting to find a better result for RAR(3).

From the algorithmic perspective, it remains open whether any of the studied problems and RAI in particular admits an approximation algorithm with a rate smaller than 2. There have been some results [32, 27] for RAI using promising linear programming relaxations that may be useful in this context. Another possibility is the application of the local search techniques originally used by Svensson [30] for the restricted assignment problem. This approach recently yielded a breakthrough for the graph balancing problem [14].

Finally, while a PTAS for RAR(1) is known [24], it is unclear whether the problem admits a so called *efficient* PTAS with a running time of the form $f(1/\varepsilon)\text{poly}(|I|)$ for some computable function f.

References

1 Aditya Bhaskara, Ravishankar Krishnaswamy, Kunal Talwar, and Udi Wieder. Minimum makespan scheduling with low rank processing times. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 937–947, 2013. doi:10.1137/1.9781611973105.67.

2 Andreas Brandstädt and Vadim V. Lozin. On the linear structure and clique-width of bipartite permutation graphs. *Ars Comb.*, 67, 2003.

3 Deeparnab Chakrabarty and Kirankumar Shiragur. Graph balancing with two edge types. *CoRR*, abs/1604.06918, 2016. arXiv:1604.06918.

4 Lin Chen, Klaus Jansen, and Guochuan Zhang. On the optimality of exact and approximation algorithms for scheduling problems. *J. Comput. Syst. Sci.*, 96:1–32, 2018. doi:10.1016/j.jcss.2018.03.005.

5 Lin Chen, Dániel Marx, Deshi Ye, and Guochuan Zhang. Parameterized and approximation results for scheduling with a low rank processing time matrix. In *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, pages 22:1–22:14, 2017. doi:10.4230/LIPIcs.STACS.2017.22.

6 Lin Chen, Deshi Ye, and Guochuan Zhang. An improved lower bound for rank four scheduling. *Oper. Res. Lett.*, 42(5):348–350, 2014. doi:10.1016/j.orl.2014.06.003.

7 Tomás Ebenlendr, Marek Krcál, and Jirí Sgall. Graph balancing: A special case of scheduling unrelated parallel machines. *Algorithmica*, 68(1):62–80, 2014. doi:10.1007/s00453-012-9668-9.

8 Leah Epstein and Asaf Levin. Scheduling with processing set restrictions: Ptas results for several variants. *International Journal of Production Economics*, 133(2):586–595, 2011. doi:10.1016/j.ijpe.2011.04.024.

9 Pinar Heggernes and Dieter Kratsch. Linear-time certifying recognition algorithms and forbidden induced subgraphs. *Nord. J. Comput.*, 14(1-2):87–108, 2007.

10 Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, 34(1):144–162, 1987. doi:10.1145/7531.7535.

11 Ellis Horowitz and Sartaj Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *J. ACM*, 23(2):317–327, 1976. doi:10.1145/321941.321951.

12 Chien-Chung Huang and Sebastian Ott. A combinatorial approximation algorithm for graph balancing with light hyper edges. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 49:1–49:15, 2016. doi:10.4230/LIPIcs.ESA.2016.49.

13 Klaus Jansen, Marten Maack, and Roberto Solis-Oba. Structural parameters for scheduling with assignment restrictions. In *Algorithms and Complexity - 10th International Conference, CIAC 2017, Athens, Greece, May 24-26, 2017, Proceedings*, pages 357–368, 2017. doi:10.1007/978-3-319-57586-5_30.

14 Klaus Jansen and Lars Rohwedder. Local search breaks 1.75 for graph balancing. *CoRR*, abs/1811.00955, 2018. arXiv:1811.00955.

15 Kamyar Khodamoradi. *Algorithms for Scheduling and Routing Problems*. PhD thesis, Simon Fraser University, 2016.

16 Kamyar Khodamoradi, Ramesh Krishnamurti, Arash Rafiey, and Georgios Stamoulis. PTAS for ordered instances of resource allocation problems. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2013, December 12-14, 2013, Guwahati, India*, pages 461–473, 2013. doi:10.4230/LIPIcs.FSTTCS.2013.461.

17 Kamyar Khodamoradi, Ramesh Krishnamurti, Arash Rafiey, and Georgios Stamoulis. PTAS for ordered instances of resource allocation problems with restrictions on inclusions. *CoRR*, abs/1610.00082, 2016. arXiv:1610.00082.

18 Kangbok Lee, Joseph Y.-T. Leung, and Michael L. Pinedo. Makespan minimization in online scheduling with machine eligibility. *Annals OR*, 204(1):189–222, 2013. doi:10.1007/s10479-012-1271-6.

19 Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990. doi:10.1007/BF01585745.

20 Joseph Y-T Leung and Chung-Lun Li. Scheduling with processing set restrictions: A survey. *International Journal of Production Economics*, 116(2):251–262, 2008. doi:10.1016/j.ijpe.2008.09.003.

21 Joseph Y-T Leung and Chung-Lun Li. Scheduling with processing set restrictions: A literature update. *International Journal of Production Economics*, 175:1–11, 2016. doi:10.1016/j.ijpe.2014.09.038.

22 Marten Maack and Klaus Jansen. Inapproximability results for scheduling with interval and resource restrictions. *CoRR*, abs/1907.03526, 2019. URL: http://arxiv.org/abs/1907.03526.

23 Gabriella Muratore, Ulrich M. Schwarz, and Gerhard J. Woeginger. Parallel machine scheduling with nested job assignment restrictions. *Oper. Res. Lett.*, 38(1):47–50, 2010. doi:10.1016/j.orl.2009.09.010.

24 Jinwen Ou, Joseph Y-T Leung, and Chung-Lun Li. Scheduling parallel machines with inclusive processing set restrictions. *Naval Research Logistics (NRL)*, 55(4):328–338, 2008. doi:10.1002/nav.20286.

25 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226, 1978. doi:10.1145/800133.804350.

26 Petra Schuurman and Gerhard J Woeginger. Polynomial time approximation algorithms for machine scheduling: Ten open problems. *Journal of Scheduling*, 2(5):203–213, 1999. doi:10.1002/(SICI)1099-1425(199909/10)2:5<203::AID-JOS26>3.0.CO;2-5.

27 Ulrich M. Schwarz. *Approximation algorithms for scheduling and two-dimensional packing problems*. PhD thesis, University of Kiel, 2010. URL: http://eldiss.uni-kiel.de/macau/receive/dissertation_diss_00005147.

28 Ulrich M. Schwarz. A PTAS for scheduling with tree assignment restrictions. *CoRR*, abs/1009.4529, 2010. arXiv:1009.4529.

29 Georgios Stamoulis. Private communication, 2019.

30 Ola Svensson. Santa claus schedules jobs on unrelated machines. *SIAM J. Comput.*, 41(5):1318–1341, 2012. doi:10.1137/110851201.

31 Craig A. Tovey. A simplified np-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984. doi:10.1016/0166-218X(84)90081-7.

32 Chao Wang and René Sitters. On some special cases of the restricted assignment problem. *Inf. Process. Lett.*, 116(11):723–728, 2016. doi:10.1016/j.ipl.2016.06.007.

33 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. URL: http://www.cambridge.org/de/knowledge/isbn/item5759340/?site_locale=de_DE.