Complexity and Inapproximability Results for Parallel Task Scheduling and Strip Packing

Sören Henning¹ · Klaus Jansen¹ · Malin Rau¹ · Lars Schmarje¹

Abstract
We study Parallel Task Scheduling $Pm|size_j|C_{\text{max}}$ with a constant number of machines. This problem is known to be strongly NP-complete for each $m \geq 5$, while it is solvable in pseudo-polynomial time for each $m \leq 3$. We give a positive answer to the long-standing open question whether this problem is strongly NP-complete for $m = 4$. As a second result, we improve the lower bound of $\frac{11}{12}$ for approximating pseudo-polynomial Strip Packing to $\frac{5}{4}$. Since the best known approximation algorithm for this problem has a ratio of $\frac{5}{4} + \varepsilon$, this result almost closes the gap between approximation ratio and inapproximability result. Both results are proved by a reduction from the strongly NP-complete problem 3-Partition.

Keywords Parallel Task Scheduling · Strip Packing · 3-Partition · Inapproximability · Complexity

This article is part of the Topical Collection on Computer Science Symposium in Russia (2018)

This work was supported by German Research Foundation (DFG) project JA 612/20-1

Malin Rau
mra@informatik.uni-kiel.de

Sören Henning
she@informatik.uni-kiel.de

Klaus Jansen
kj@informatik.uni-kiel.de

Lars Schmarje
las@informatik.uni-kiel.de

¹ Institut für Informatik, Christian-Albrechts-Universität zu Kiel, Kiel, Germany
1 Introduction

In the Parallel Task Scheduling problem denoted as \( P|\text{size}_j|C_{\text{max}} \) in the three-field-notation, a set of jobs \( J \) has to be scheduled on \( m \) machines minimizing the makespan \( C_{\text{max}} \). Each job \( j \in J \) has a processing time \( p(j) \in \mathbb{N} \) and requires \( q(j) \in \mathbb{N} \) machines. A schedule \( S \) is given by two functions \( \sigma: J \rightarrow \mathbb{N} \) and \( \rho: J \rightarrow 2^{\{1,\ldots,m\}} \). The function \( \sigma \) maps each job to a start point in the schedule, while \( \rho \) maps each job to the set of machines it is processed on. We say a machine \( i \) contains a job \( j \in J \) if \( i \in \rho(j) \). A schedule is feasible if each machine processes at most one job at a time and each job is processed on the required number of machines (i.e. \( |\rho(j)| = q(j) \)). The objective is to find a feasible schedule \( S \) minimizing the makespan \( C_{\text{max}} := \max_{j \in J} (\sigma(j) + p(j)) \).

In 1989, Du and Leung [9] proved the Parallel Task Scheduling problem \( P|\text{size}_j|C_{\text{max}} \) to be strongly NP-complete for all \( m \geq 5 \), while \( P|\text{size}_j|C_{\text{max}} \) is solvable by a pseudo-polynomial time algorithm for all \( m \leq 3 \). In this paper, we address the case of \( m = 4 \), which has been open since and prove:

**Theorem 1** Parallel Task Scheduling on 4 machines is strongly NP-complete.

Building on this result, we can prove a lower bound for the absolute approximation ratio of pseudo-polynomial time algorithms for the Strip Packing problem, where an algorithm \( A \) has absolute approximation ratio \( \alpha \) if \( A(I)/\text{OPT}(I) \leq \alpha \) for all instances \( I \) of the given problem. In the Strip Packing problem a set of rectangular items \( I \) has to be placed into a strip of width \( W \in \mathbb{N} \) and infinite height. Each item \( i \in I \) has a width \( w(i) \in \mathbb{N}_{\leq W} \) and a height \( h(i) \in \mathbb{N} \). A packing of the items \( I \) into the strip is a function \( \rho: I \rightarrow \mathbb{Q}_0 \times \mathbb{Q}_0 \), which places the items axis-parallel into the strip by assigning the left bottom corner of an item to a position in the strip such that for each item \( i \in I \) with \( \rho(i) = (x_i, y_i) \) we have \( x_i + w(i) \leq W \). We say two items \( i, j \in I \) overlap if they share an inner point. A packing is feasible if no two items overlap. The height of a packing is defined as \( H := \max_{i \in I} y_i + h(i) \). The objective is to find a feasible packing of the items \( I \) into the strip that minimizes the packing height. If all item sizes are integral, we can transform feasible packings to packings where all positions are integral without enlarging the packing height [5]. Therefore, we can assume that we have packings of the form \( \rho: I \rightarrow \mathbb{N}_0 \times \mathbb{N}_0 \).

Lately, pseudo-polynomial time algorithms for Strip Packing where the width of the strip is allowed to appear polynomially in the running time of the algorithm, while it appears only logarithmically in the input size of the instance, gained high interest. In a series of papers [11, 20, 21, 23, 26], the best approximation ratio was improved to \( \frac{5}{4} + \varepsilon \). On the other hand, it is not possible to find an algorithm with approximation

|      | 1 | 12/11 | improvement | 5/4, [20] | [21],[11] | 7/5 + \varepsilon | 3/2 + \varepsilon |
|------|---|-------|-------------|----------|-----------|----------------|----------------|

**Fig. 1** The upper and lower bounds for the best possible approximation for pseudo-polynomial Strip Packing achieved so far

\( \varepsilon \) Springer
ratio better than $\frac{12}{11}$, except $P = NP$ [1]. In this paper, we improve this lower bound to $\frac{5}{4}$, which almost closes the gap between lower bound and best algorithm (Fig. 1).

**Theorem 2** It is NP-Hard to find a pseudo-polynomial time approximation algorithm for Strip Packing with an absolute approximation ratio strictly better than $\frac{5}{4}$.

### 1.1 Related Work

**Parallel Task Scheduling** In 1989, Du and Leung [9] proved Parallel Task Scheduling $Pm|size_j|C_{\text{max}}$ to be strongly NP-complete for all $m \geq 5$, while it is solvable by a pseudo-polynomial time algorithm for all $m \leq 3$. Amoura et al. [2], as well as Jansen and Porkolab [19], presented a polynomial-time approximation scheme (in short PTAS) for the case that $m$ is a constant. A PTAS is a family of polynomial-time algorithms that finds a solution with an approximation ratio of $(1 + \varepsilon)$ for any given value $\varepsilon > 0$. If $m$ is polynomially bounded by the number of jobs, a PTAS exists [23]. Nevertheless, if $m$ is arbitrarily large, the problem gets harder. By a simple reduction from the Partition problem, one can see that there is no polynomial time algorithm with approximation ratio smaller than $\frac{3}{2}$. Parallel Task Scheduling with arbitrarily large $m$ has been widely studied [10, 12, 25, 31]. The algorithm with the best known absolute approximation ratio of $\frac{3}{2} + \varepsilon$ was presented by Jansen [17].

**Strip Packing** The Strip Packing problem was first studied in 1980 by Baker et al. [4]. They presented an algorithm with an absolute approximation ratio of 3. This ratio was improved by a series of papers [8, 16, 27–29]. The algorithm with the best known absolute approximation ratio by Harren, Jansen, Prädel and van Stee [15] achieves a ratio of $\frac{5}{3} + \varepsilon$. By a simple reduction from the Partition problem, one can see that it is impossible to find an algorithm with better approximation ratio than $\frac{3}{2}$, unless $P = NP$.

For this problem also asymptotic approximation algorithms have been studied. An algorithm for a minimization problem has an asymptotic approximation ratio of $\alpha$, if there is a constant $c$ (which might depend on $\varepsilon$ or the maximal occurring item height $h_{\text{max}}$) such that the objective value $A(I)$ computed by the algorithm is bounded by $\alpha OPT(I) + c$. The lower bound of $\frac{3}{2}$ does not hold for asymptotic approximation ratios and they have been studied in various papers [3, 8, 14]. Kenyon and Rémiла [24] presented an asymptotic fully polynomial approximation scheme (in short AF-PTAS) with additive term $O(h_{\text{max}}/\varepsilon^2)$, where $h_{\text{max}}$ is the largest occurring item height. An approximation scheme is fully polynomial if its running time is polynomial in $1/\varepsilon$ as well. This algorithm was simultaneously improved by Sviridenko [30] and Bougeret et al. [6] to an algorithm with an additive term of $O(h_{\text{max}} \log(1/\varepsilon)/\varepsilon)$. Furthermore, at the expense of the running time, Jansen and Solis-Oba [22] presented an asymptotic PTAS with an additive term of $h_{\text{max}}$.

Recently, the focus shifted to pseudo-polynomial time algorithms. Jansen and Thöle [23] presented an pseudo-polynomial time algorithm with approximation ratio of $\frac{3}{2} + \varepsilon$. Later Nadiradze and Wiese [26] presented an algorithm with ratio $\frac{7}{5} + \varepsilon$. Its
approximation ratio was independently improved to $\frac{7}{4} + \varepsilon$ by Gálvez et al. [11] and by Jansen and Rau [21]. $5/4 + \varepsilon$ is the best approximation ratio so far, achieved by an algorithm by Jansen and Rau [20]. All these algorithms have a polynomial running time if the width of the strip $W$ is bounded by a polynomial in the number of items.

In contrast to Parallel Task Scheduling, Strip Packing cannot be approximated arbitrarily close to 1, if we allow pseudo-polynomial running time. This was proved by Adamaszek et al. [1] by presenting a lower bound of $\frac{12}{11}$. As a consequence, Strip Packing admits no quasi-polynomial time approximation scheme, unless NP $\subseteq$ DTIME($2^{\text{polylog}(n)}$). For an overview on 2-dimensional packing problems and open questions regarding these problems, we refer to the survey by Christensen et al. [7].

1.2 Organization of this Paper

In Section 2, we will prove Theorem 1 by a reduction from the strongly NP-complete problem 3-Partition. First, we describe the jobs to construct for this reduction. Afterward, we prove: if the 3-Partition instance is a Yes-instance, then there is a schedule with a specific makespan, and if there is a schedule with this specific makespan then the 3-Partition instance has to be a Yes-instance. While the first claim can be seen directly, the proof of the second claim is more involved. Proving the second claim, we first show that it can be w.l.o.g. supposed that each machine contains a certain set of jobs. In the next step, we prove some implications on the order in which the jobs appear on the machines which finally leads to the conclusion that the 3-Partition instance has to be a Yes-instance. In Section 3 we discuss the implications for the inapproximability of pseudo-polynomial Strip Packing.

2 Hardness of Scheduling Parallel Tasks

In this Section, we prove Theorem 1 by a reduction from the 3-Partition problem. In this problem, we are given a list $I = (\iota_1, \ldots, \iota_{3z})$ of $3z$ positive integers with $\sum_{i=1}^{3z} \iota_i = zD$ and $D/4 < \iota_i < D/2$ for each $1 \leq i \leq 3z$. The problem is to decide whether there exists a partition of the set $I = \{1, \ldots, 3z\}$ into sets $I_1, \ldots, I_z$ such that $\sum_{i \in I_j} \iota_i = D$ for each $1 \leq j \leq z$. This problem is strongly NP-complete see [13] problem [SP15]. Hence, it cannot be solved in pseudo-polynomial time, unless P = NP.

Before we start constructing the reduction, we introduce some notations. Let $j \in J$ and $J' \subseteq J$. We define the work of $j$ as $w(j) := p(j) \cdot \rho(j)$ and the total work of $J'$ as $w(J') := \sum_{j \in J'} w(j)$. For a given schedule $S = (\sigma, \rho)$, we denote by $n_j(J')$ the number of jobs from the set $J'$ that are finished before the start of the job $j$, i.e., $n_j(J') = |\{i \in J' : \sigma(i) + p(i) \leq \sigma(j)\}|$. Furthermore, we will use a notation defined in [9] for swapping a part of the content of two machines; let $j \in J$ be a job that is processed by at least two machines, $M$ and $M'$, with start point $\sigma(j)$, i.e., $n_j(J') = \{|i \in J' : \sigma(i) + p(i) \leq \sigma(j)\}|$. We can swap the content of the machines $M$ and $M'$ after time $\sigma(j)$ without violating any scheduling constraint. We define this swapping operation as SWAP($\sigma(j), M, M'$).

The main idea of our reduction is to construct a set of structure jobs. These structure jobs have the property that each possible way to schedule them with the optimal makespan leaves $z$ gaps, each with processing time $D$, i.e., it happens exactly at $z$
Fig. 2 Packing of structure jobs with gaps (hatched area) for 3-Partition items. The items in the green area (left) are repeated \( z \) times. With the displayed choice of processing times, the items in the red area (right) can be rotated by 180 degrees such that \( \alpha \) is scheduled on \( M_4 \) after the job in \( B \) and \( \beta \) is scheduled on \( M_1 \) before the job in \( A \) distinct times that a machine is idle, and the duration of each idle time is exactly \( D \), see Fig. 2 at the hatched areas. As a consequence, partition jobs, which have processing times equal to the 3-Partition numbers, can only be scheduled with the desired makespan if the 3-Partition instance is a Yes-instance.

2.1 Construction

In this section, we will construct a scheduling instance for \( P_4|\text{size}||C_{\text{max}} \) from a given 3-Partition instance. In the following two paragraphs, we will give an intuition which jobs we introduce why with which processing time. An overview of the introduced jobs and their processing times can be found in Table 1 and the fourth paragraph of this section.

Given a 3-Partition instance, we construct ten disjoint sets of jobs \( A, B, a, b, c, \alpha, \beta, \gamma, \delta, \) and \( \lambda \), which will be forced to be scheduled as in Fig. 2 by choosing suitable processing times. In the first step, we add a unique token to the processing time of each set of jobs to be processed simultaneously to ensure that these jobs have to be processed at the same time in every schedule. As this token, we choose \( D_x \), where \( x \in \{2, \ldots, 7\} \) and \( D \) is the required sum of the items in each partition set.

### Table 1 Overview of the generated jobs

| \( q(j) \) | \( p(j) \) | \( p_A \) | \( p_B \) | \( p_a \) | \( p_b \) | \( p_\alpha \) | \( p_\beta \) | \( p_\gamma \) | \( p_\delta \) | \( p_\lambda \) | \( p_\iota \) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | \( D^4 \) | \( A \) | \( B \) | \( a \) | \( b \) | \( \alpha \) | \( \beta \) | \( \gamma \) | \( \delta \) | \( \lambda \) | \( P \) |
| 3 | \( D^2 \) | if \( j \in A \) | \( := \{A_0, \ldots, A_z\} \) | if \( j \in B \) | \( := \{B_0, \ldots, B_z\} \) | if \( j \in a \) | \( := \{a_1, \ldots, a_z\} \) | if \( j \in b \) | \( := \{b_1, \ldots, b_z\} \) | if \( j = c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) |
| 2 | \( D^4 + 3D^6 + 3zD^8 \) | if \( j \in A \) | \( := \{A_0, \ldots, A_z\} \) | if \( j \in b \) | \( := \{b_1, \ldots, b_z\} \) | if \( j = c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) | if \( j \in a \) | \( := \{a_1, \ldots, a_z\} \) |
| 2 | \( D^4 + D^7 + 3zD^8 \) | if \( j \in B \) | \( := \{B_0, \ldots, B_z\} \) | if \( j = c_i \in a \) | \( := \{a_1, \ldots, a_z\} \) | if \( j = c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) |
| 2 | \( D^4 + (z + i)D^8 \) | if \( j \in a \) | \( := \{a_1, \ldots, a_z\} \) | if \( j = c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) | if \( j = c_i \in a \) | \( := \{a_1, \ldots, a_z\} \) |
| 1 | \( D^4 + D^7 + 4zD^8 \) | if \( j \in a \) | \( := \{a_1, \ldots, a_z\} \) | if \( j = c_i \in a \) | \( := \{a_1, \ldots, a_z\} \) | if \( j \in c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) | if \( j \in B \) | \( := \{b_1, \ldots, b_z\} \) |
| 1 | \( D^4 + D^7 + 4zD^8 \) | if \( j \in B \) | \( := \{b_1, \ldots, b_z\} \) | if \( j = c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) | if \( j \in b \) | \( := \{b_1, \ldots, b_z\} \) | if \( j = c_i \in a \) | \( := \{a_1, \ldots, a_z\} \) |
| 1 | \( D^4 + (3z - i)D^8 \) | if \( j \in a \) | \( := \{a_1, \ldots, a_z\} \) | if \( j = c_i \in a \) | \( := \{a_1, \ldots, a_z\} \) | if \( j \in b \) | \( := \{b_1, \ldots, b_z\} \) | if \( j = c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) |
| 1 | \( D^4 + (3z - i)D^8 \) | if \( j \in b \) | \( := \{b_1, \ldots, b_z\} \) | if \( j = c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) | if \( j \in a \) | \( := \{a_1, \ldots, a_z\} \) | if \( j = c_i \in a \) | \( := \{a_1, \ldots, a_z\} \) |
| 1 | \( D^4 + 3D^6 + zD^8 \) | if \( j \in b \) | \( := \{b_1, \ldots, b_z\} \) | if \( j = c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) | if \( j \in a \) | \( := \{a_1, \ldots, a_z\} \) | if \( j = c_i \in a \) | \( := \{a_1, \ldots, a_z\} \) |
| 1 | \( D^4 + 3D^6 + 2zD^8 \) | if \( j \in b \) | \( := \{b_1, \ldots, b_z\} \) | if \( j = c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) | if \( j \in a \) | \( := \{a_1, \ldots, a_z\} \) | if \( j = c_i \in a \) | \( := \{a_1, \ldots, a_z\} \) |
| 1 | \( D^4 + (3z - i)D^8 - D \) | if \( j \in B \) | \( := \{b_1, \ldots, b_z\} \) | if \( j = c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) | if \( j \in b \) | \( := \{b_1, \ldots, b_z\} \) | if \( j = c_i \in a \) | \( := \{a_1, \ldots, a_z\} \) |
| 1 | \( D^4 + (3z - i)D^8 \) | if \( j \in B \) | \( := \{b_1, \ldots, b_z\} \) | if \( j = c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) | if \( j = c_i \in a \) | \( := \{a_1, \ldots, a_z\} \) | if \( j \in a \) | \( := \{a_1, \ldots, a_z\} \) |
| 1 | \( D^4 + 2zD^8 \) | if \( j \in b \) | \( := \{b_1, \ldots, b_z\} \) | if \( j = c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) | if \( j \in a \) | \( := \{a_1, \ldots, a_z\} \) | if \( j = c_i \in a \) | \( := \{a_1, \ldots, a_z\} \) |
| 1 | \( \iota_j \) | if \( j \in P \) | \( := \{p_1, \ldots, p_{3z}\} \) | if \( j \in a \) | \( := \{a_1, \ldots, a_z\} \) | if \( j \in b \) | \( := \{b_1, \ldots, b_z\} \) | if \( j = c_i \in c \) | \( := \{c_0, \ldots, c_z\} \) |
For example for jobs in $B$ we define a processing time of $D^2$, while we define the processing time of each job in $\alpha$ such that it contains $D^7 + D^2 + D^3$, see Fig. 2.

Unfortunately, the tokens $D^2$ to $D^7$ are not enough to ensure that the schedule in Fig. 2 is the only possible one. Consider the jobs contained in the red area (right) in Fig. 2. With the choice of processing times as shown in the figure, it is possible to rotate the red area by 180 degrees such that $\alpha$ is scheduled on $M_4$ and $\beta$ is scheduled on $M_1$. After rotating every second of these set of jobs, it is possible to reorder the areas for the 3-Partition items into two or three areas, see Fig. 3. To prohibit this possibility to rotate, we introduce one further token $D^8$. This token is added to the processing time of some jobs such that the combined processing time of the jobs in the red area on $M_1$ differs from the one on $M_4$. To ensure this, we have to give up the property that in each of the sets $A, B, a, b, c, \alpha, \beta, \gamma, \delta$ all jobs have the same processing time. More precisely, each job in the sets $c, \delta, \gamma$ receives a unique processing time.

In the following, we describe the jobs constructed for the reduction. We introduce two sets $A$ and $B$ of 3-processor jobs, three sets $a, b$ and $c$ of 2-processor jobs, and five sets $\alpha, \beta, \gamma, \delta, \lambda$ of 1-processor jobs. The description of the jobs inside these sets and their processing times can be found in Table 1. We call these jobs structure jobs. Additionally, we generate for each $i \in \{1, \ldots, 3z\}$ one 1-processor job, called partition job, with processing time $i$ and define $P$ as the set containing all partition jobs. Last, we define $W := (z + 1)(D^2 + D^3 + D^4) + z(D^5 + D^6 + D^7) + z(7z + 1)D^8$. Note that the total work of the introduced jobs adds up to $4W$, i.e., a schedule without idle times has makespan $W$ while each schedule containing idle times has a makespan, which is strictly larger than $W$.

Given a set $J \subseteq \bigcup\{A, B, a, b, c, \alpha, \beta, \delta, \lambda\}$ of the jobs constructed this way, their total processing time $p(J)$ has the form $\sum_{i=2}^{7} x_i D^i$, with $x_i \in \mathbb{N}$ for $i = 2, \ldots, 7$. For each occurring $x_i$, we want the tokens $D^i$ to be unique in the way that $x_i D^i < D^{i+1}$ for each possible occurring sum of processing times of structure jobs and each $i = 2, \ldots, 7$. Let $k_{\text{max}}$ be the largest occurring coefficient in the sum of processing times of any given subset of the generated structure jobs, i.e., $k_{\text{max}} \leq 4z(7z + 1)$. If $D \leq k_{\text{max}}$, we scale each number in the 3-Partition instance with $k_{\text{max}}$, before constructing the jobs. As a result in the scaled instance it holds that $k_{\text{max}} D^i < D^{i+1}$.

Since $k_{\text{max}}$ depends polynomially on $z$, the input size of the scaled instance will still depend polynomially on the input size of the original instance. In the following, let us assume that $D > k_{\text{max}}$ in the given 3-Partition instance. Note that in a schedule without idle times, a machine cannot contain a set of jobs, with processing times that

![Fig. 3](image-url) A reordering we have to prohibit, because it fuses the areas for 3-Partition items into two areas, one area on $M_2$ and one area on $M_3$ if $z$ is even, and into three areas if $z$ is odd.
add up to a value where one of the coefficients is larger than the corresponding one in $W$.

In the following two subsections, we will prove that there is a schedule with makespan $W$ if and only if the 3-Partition instance is a Yes-instance.

### 2.2 Partition to Schedule

Let $\mathcal{I}$ be a Yes-instance of 3-Partition with partition $I_1, \ldots, I_z$. One can easily verify that the structure jobs can be scheduled as shown in Fig. 4. After each job $\gamma_j$, for each $1 \leq j \leq z$, we have a gap with processing time $D$. We schedule the partition jobs with indices out of $I_j$ directly after $\gamma_j$. Their processing times add up to $D$, and therefore they fit into the gap. The resulting schedule has a makespan of $W$.

### 2.3 Schedule to Partition

Let a schedule $S = (\sigma, \rho)$ with makespan $W$ be given. We will now step by step describe why $\mathcal{I}$ has to be a Yes-instance of 3-Partition. In the first step, we will show that we can transform the schedule such that each machine contains a certain set of jobs.

**Lemma 1** We can transform the schedule $S$ into a schedule such that $M_1$ contains the jobs $\bigcup \{A, a, \alpha, \lambda_1\}$, $M_2$ contains the jobs $\bigcup \{A, B, c, \dot{a}, \dot{b}, \dot{\gamma}, \dot{\delta}\}$, $M_3$ contains the jobs $\bigcup \{A, B, c, \bar{a}, \bar{b}, \bar{\gamma}, \bar{\delta}\}$, and $M_4$ contains the jobs $\bigcup \{B, b, \beta, \lambda_2\}$, where $a = \dot{a} \cup \bar{a}$, $b = \dot{b} \cup \bar{b}$, $\gamma = \dot{\gamma} \cup \bar{\gamma}$, and $\delta = \dot{\delta} \cup \bar{\delta}$. Furthermore, if the jobs are scheduled in this way, it holds that $|\dot{a}| = |\dot{\gamma}|$ and $|\dot{b}| = |\dot{\delta}|$.

**Proof** First, we will show that the content of the machines can be swapped, without enlarging the makespan, such that $M_2$ and $M_3$ each contain all the jobs in $A \cup B$. We will show this claim inductively. For the induction basis, consider the job in $A \cup B$ with the smallest starting point in this set. We can swap the complete content of the machines such that $M_2$ and $M_3$ contain this job. For the induction step, let us assume that the first $i$ jobs from the set $A \cup B$ are scheduled on the machines $M_2$ and $M_3$. Consider the $(i + 1)$st job. This job is either already scheduled on the machines $M_2$ and $M_3$, and we do nothing, or there is one machine $M \in \{M_2, M_3\}$, which does not contain this job. Let us assume the latter. Let $M' \in \{M_1, M_4\}$ be the third machine

![Fig. 4 An optimal schedule, for a Yes-instance, where $t_0 := \sum_{k=2}^{4} D^k + zD^8$, $t_1 := \sum_{k=2}^{7} D^k + (7z - 1)D^8$, and $t_2 := \sum_{k=2}^{7} D^k + 7zD^8$](image-url)
containing the $i$-th job in $A \cup B$. We transform the schedule such that $M_2$ and $M_3$ contain the $(i+1)$-th job, by performing a swapping operation $\text{SWAP}(\sigma(x_i), M, M')$. After this swap $M$, and hence both machines $M_2$ and $M_3$, will contain the $(i+1)$st job, which concludes the proof that the content of the machines can be swapped such that $M_2$ and $M_3$ each contain all the jobs in $A \cup B$.

In the next step, we will determine the set of jobs contained by the machines $M_1$ and $M_4$ using the token $D^8$. Besides the jobs in $A \cup B$, $M_2$ and $M_3$ contain jobs with total processing time of $(z+1)D^3 + zD^5 + zD^6 + zD^7 + z(7z+1)D^8$. Hence, $M_2$ and $M_3$ cannot contain jobs in $\alpha \cup \beta \cup \lambda$, since their processing times contain $D^2$ or $D^4$. Therefore, each job in $\alpha \cup \beta \cup \lambda$ has to be processed either on $M_1$ or on $M_4$. Furthermore, each job in $A \cup B$ has to be processed on one of the machines $M_1$ or $M_4$ additional to the machines $M_2$ and $M_3$ since each of these jobs needs three machines to be scheduled. In addition to the jobs in $\bigcup \{A, B, \alpha, \beta, \lambda\}$, $M_1$ and $M_4$ together contain further jobs with a total processing time of $zD^5 + 2zD^6 + zD^7 + 6z^2D^8$. Exclusively jobs from the set $a \cup b$ have a processing time containing $D^6$. Therefore, each machine processes $z$ of them. Hence corresponding to $D^8$, a total processing time of $3z^2D^8$ is used by jobs in the set $a \cup b$ on each machine. This leaves a processing time of $(4z^2 + z)D^8$ for the jobs in $\alpha \cup \beta \cup \lambda$ on $M_1$ and $M_4$. All the $2(z + 1)$ jobs in $\alpha \cup \beta \cup \lambda$ contain $D^3$ in their processing time. Therefore, each machine $M_1$ and $M_4$ processes exactly $z + 1$ of them. We will swap the content of $M_1$ and $M_4$ so that $\lambda_1$ is scheduled on $M_1$. As a consequence, $M_1$ processes $z$ jobs from the set $\alpha \cup \beta \cup \{\lambda_2\}$, with processing times that sum up to $4z^2D^8$ in the $D^8$ component. The jobs in $\alpha$ have with $4zD^8$ the largest amount of $D^8$ in their processing time. Therefore, $M_1$ has to process all of them since $z \cdot 4zD^8 = 4z^2D^8$, while $M_4$ contains the jobs in $\beta \cup \{\lambda_2\}$. Since we have $p(\alpha \cup \{\lambda_1\}) = (z + 1)D^2 + (z + 1)D^3 + zD^7 + z(4z + 1)D^8$, jobs from the set $A \cup B \cup a \cup b$ with total processing time of $(z + 1)D^4 + zD^5 + zD^6 + 3z^2D^8$ have to be scheduled on $M_1$. In this set, the jobs in $A$ are the only jobs with processing times containing $D^4$, while the jobs in $a$ are the only jobs with a processing time containing $D^5$. As a consequence, $M_1$ processes the jobs $\bigcup \{A, a, \alpha, \{\lambda_1\}\}$. Analogously we can deduce that $M_4$ processes the jobs $\bigcup \{B, b, \beta, \{\lambda_2\}\}$.

In the last step, we will determine which jobs are scheduled on $M_2$ and $M_3$. As shown before, each of them contains the jobs $A \cup B$. Furthermore, since no job in $c$ is scheduled on $M_1$ or $M_4$, and they require two machines to be processed, machines $M_2$ and $M_3$ both contain the set $c$. Additionally, each job in $\gamma \cup \delta$ has to be scheduled on $M_2$ or $M_3$ since they are not scheduled on $M_1$ or $M_4$. Each job in $a \cup b$ occupies one of the machines $M_1$ and $M_4$. The second machine they occupy is either $M_2$ or $M_3$. Let $\hat{a} \subseteq a$ be the set of jobs that is scheduled on $M_2$ and $\hat{a} \subseteq a$ be the set that is scheduled on $M_3$. Clearly $a = \hat{a} \cup \hat{a}$. We define the sets $\hat{b}, \hat{b}, \hat{\delta}, \hat{\delta}, \hat{\gamma}$, and $\hat{\gamma}$ analogously. By this definition, $M_2$ contains the jobs $\bigcup \{A, B, \hat{a}, \hat{b}, \hat{\delta}, \hat{\gamma}, c\}$ and $M_3$ contains the jobs $\bigcup \{A, B, \hat{a}, \hat{b}, \hat{\delta}, \hat{\gamma}, c\}$.

We still have to prove that $|\hat{a}| = |\hat{b}|$ and $|\hat{b}| = |\hat{\delta}|$. First, we notice that $|\hat{a}| + |\hat{b}| = z$ since these jobs are the only jobs with a processing time containing $D^6$. So besides the jobs in $\bigcup \{A, B, c, \hat{a}, \hat{b}\}$, $M_2$ contains jobs with total processing time of $(z - |\hat{a}|)D^5 + (z - |\hat{b}|)D^7 + \sum_{i=1}^{5}(3z - i)D^8 = |\hat{b}|D^5 + |\hat{a}|D^7 + \sum_{i=1}^{5}(3z - i)D^8$.
Since the jobs in $\delta$ are the only jobs in $\delta \cup \gamma$ having a processing time containing $D^5$, we have $|\delta| = |\hat{b}|$ and analogously $|\hat{\gamma}| = |\hat{a}|$.

In the next steps, we will prove that it is possible to transform the order in which the jobs appear on the machines to the one in Fig. 4. Notice that, since there is no idle time in the schedule, each start point of a job $i$ is given by the sum of processing times of the jobs on the same machine scheduled before $i$. So the start position $\sigma(i)$ of a job $i$ has the form

$$\sigma(i) = x_0 + x_2 D^2 + x_3 D^3 + x_4 D^4 + x_5 D^5 + x_6 D^6 + x_7 D^7 + x_8 D^8$$

for $-z D \leq x_0 \leq z D < D^2$ and $0 \leq x_j \leq 4z(7z + 1) \leq D$ for each $2 \leq j \leq 8$. Note that $-z D \leq x_0$ since the processing time of the jobs in $\gamma$ is given by $D^7 + (3z - i)D^8 - D$ and there are at most $z$ of them while $x_0 \leq z D$ since the total sum of processing times of partition jobs is at most $z D$. This equation for $\sigma(i)$ allows us to analyze how many jobs of which type are scheduled before a job $i$ on the machine that processes $i$. For example, let us look at the coefficient $x_4$. This value is just influenced by jobs with processing times containing $D^4$. The only jobs with these processing times are the jobs in the set $A \cup \beta \cup \lambda_2$. The jobs in $\beta \cup \lambda_2$ are just processed on $M_4$, while the jobs in $A$ each are processed on the three machines $M_1$, $M_2$, and $M_3$. Therefore, we know that at the starting point $\sigma(i)$ of a job $i$ scheduled on machines $M_1$, $M_2$ or $M_3$ we have that $x_4 = n_i(A)$. Furthermore, if $i$ is scheduled on $M_4$ we know that $x_4 = n_i(\beta) + n_i(\lambda_2)$. In Table 2, we present which sets influences which coefficients in which way when job $i$ is started on the corresponding machine.

Let us consider the start point $\sigma(i)$ of a job $i$ that uses more than one machine. We know that $\sigma(i)$ is the same on all the used machines and therefore the coefficients are the same as well. In the following, we will study for each of the sets $A$, $B$, $a$, $b$, $c$ what we can conclude for the starting times of these jobs. For each of the sets, we will present an equation, which holds at the start of each item in this set. These equations give us a strong set of tools for our further arguing.

**Lemma 2** Let be $A' \in A$, $B' \in B$, $a' \in a$, $b' \in b$, and $c' \in c$. It holds that

\begin{align*}
  n_{A'}(c) - n_{A'}(\lambda_1) &= n_{A'}(B) - n_{A'}(\lambda_1) = n_{A'}(\alpha) = n_{A'}(b) = n_{A'}(a), \\
  n_{B'}(c) - n_{B'}(\lambda_2) &= n_{B'}(A) - n_{B'}(\lambda_2) = n_{B'}(\beta) = n_{B'}(a) = n_{B'}(b).
\end{align*}

| $M_1$          | $M_2$          | $M_3$          | $M_4$          |
|----------------|----------------|----------------|----------------|
| $n_i(\alpha)$  | $n_i(\alpha)$  | $n_i(\alpha)$  | $n_i(\alpha)$  |
| $n_i(\alpha')$ | $n_i(\beta)$   | $n_i(\beta)$   | $n_i(\beta)$   |
| $n_i(\gamma)$  | $n_i(\gamma)$  | $n_i(\gamma)$  | $n_i(\gamma)$  |

Table 2 Overview of the values of the coefficients at the start point of a job $i$, if $i$ is scheduled on machine $M_j$.
\[ n_{a'}(B) = n_{a'}(\alpha) + n_{a'}([\lambda_1]) = n_{a'}(c), \]  
\[ n_{b'}(A) = n_{b'}(\beta) + n_{b'}([\lambda_2]) = n_{b'}(c), \]

and

\[ n_{c'}(b) = n_{c'}(a). \]

**Proof** We will prove these equations using the conditions for the coefficients from Table 2. We write \( x_i \) when the coefficient \( x_i \) is the reason why the equality is true.

To prove (1), we will consider the start points of the jobs in \( A \). Each job \( A' \in A \) is scheduled on machines \( M_1, M_2 \) and \( M_3 \). As a consequence the coefficients on all these tree machines have to be the same, when the job \( A' \) starts. Therefore, we know that at \( \sigma(A') \) we have

\[ n_{A'}(B) = x_2 n_{A'}(\alpha) + n_{A'}([\lambda_1]) = x_3 n_{A'}(c). \]  

(i)

Furthermore, we know that

\[ n_{A'}(a) = x_6 n_{A'}(\tilde{a}) + n_{A'}(\tilde{b}) = x_6 n_{A'}(\tilde{a}) + n_{A'}(\hat{b}). \]

Since \( n_{A'}(a) = n_{A'}(\tilde{a}) + n_{A'}(\hat{a}) \) and \( n_{A'}(b) = n_{A'}(\tilde{b}) + n_{A'}(\hat{b}) \), because \( a = \tilde{a} \cup \hat{a} \) and \( b = \tilde{b} \cup \hat{b} \), we can deduce that \( n_{A'}(\tilde{a}) = n_{A'}(\tilde{b}) \) and \( n_{A'}(\hat{a}) = n_{A'}(\hat{b}) \) and therefore

\[ n_{A'}(a) = n_{A'}(b). \]  

(ii)

Additionally, we know that

\[ n_{A'}(\alpha) = x_7 n_{A'}(\tilde{b}) + n_{A'}(\tilde{\gamma}) = x_7 n_{A'}(\hat{b}) + n_{A'}(\hat{\gamma}). \]

Thanks to this equality, we can show that \( n_{A'}(\alpha) = n_{A'}(b) \). First, we show \( n_{A'}(\alpha) \geq n_{A'}(b) \). Let \( b' \in b \) be the last job in \( b \) scheduled before \( A' \) if there is any. Let us w.l.o.g assume that \( b' \in \hat{b} \). It holds that

\[ n_{A'}(b) = n_{A'}(\tilde{b}) + n_{A'}(\tilde{\gamma}) = x_7 n_{A'}(\hat{b}) + n_{A'}(\hat{\gamma}) + 1 \]

\[ \leq n_{A'}(\hat{b}) + n_{A'}(\hat{\gamma}) = x_7 n_{A'}(\alpha). \]

If there is no such \( b' \) we have \( n_{A'}(b) = 0 \leq n_{A'}(\alpha) \). Next, we show \( n_{A'}(\alpha) \leq n_{A'}(b) \). Let \( b'' \in b \) be the first job in \( b \) scheduled after \( A \) if there is any. Let us w.l.o.g assume that \( b'' \in \hat{b} \). It holds that

\[ n_{A'}(b) = n_{A'}(\tilde{b}) + n_{A'}(\tilde{\gamma}) + 1 \]

\[ \geq n_{A'}(\hat{b}) + n_{A'}(\hat{\gamma}) = x_7 n_{A'}(\alpha). \]

If there is no such \( b'' \), we have \( n_{A'}(b) = z \geq n_{A'}(\alpha) \). As a consequence, we have

\[ n_{A'}(\alpha) = n_{A'}(b). \]  

(iii)

In summary, we can deduce that

\[ n_{A'}(c) - n_{A'}([\lambda_1]) = (i) \]  
\[ n_{A'}(B) - n_{A'}([\lambda_1]) = (i) \]  
\[ n_{A'}(\alpha) = (iii) \]  
\[ n_{A'}(b) = (ii) \]  
\[ n_{A'}(a), \]

which concludes the proof of (1). Since the Table 2 is symmetrically, we can deduce correctness of (2) analogously.
Next we prove the (3) and (4). Each item $a' \in a$ is scheduled on machine $M_1$ and on one of the machines $M_2$ or $M_3$. For both possibilities $a \in \hat{a}$ or $a \in \check{a}$, we can deduce (3) directly from the Table 2:

$$n_{a'}(B) = x_2 n_{a'}(a) + n_{a'}([\lambda_1]) = x_3 n_{a'}(c).$$

Analogously, we deduce (4):

$$n_{b'}(A) = x_4 n_{b'}(\beta) + n_{b'}([\lambda_2]) = x_3 n_{b'}(c).$$

Last, we prove (5). Each item $c' \in c$ is scheduled on $M_2$ and $M_3$. Let $a' \in a$ be the job with the smallest $\sigma(a') \geq \sigma(c')$. Let us w.l.o.g assume that $a' \in \hat{a}$. It holds that

$$n_{c'}(\hat{a}) + n_{c'}(\hat{b}) = x_6 n_{c'}(\hat{a}) + n_{c'}(\hat{b}) = n_{a'}(\hat{a}) + n_{a'}(\hat{b}) = x_6 n_{a'}(a)$$

As a consequence, we have $n_{c'}(\hat{b}) \leq n_{c'}(\hat{a})$ and $n_{c'}(\hat{b}) \leq n_{c'}(\hat{a})$. Analogously, let $b' \in b$ be the job with the smallest $\sigma(b') \geq \sigma(c')$. Let us w.l.o.g assume that $b' \in \check{b}$. It holds that

$$n_{c'}(\hat{a}) + n_{c'}(\check{b}) = x_6 n_{c'}(\hat{a}) + n_{c'}(\check{b}) = n_{b'}(\check{b}) = n_{b'}(\check{b}) + n_{b'}(\check{b}).$$

Therefore, $n_{c'}(\hat{a}) \leq n_{c'}(\hat{b})$ and $n_{c'}(\hat{a}) \leq n_{c'}(\hat{b})$. As a consequence from both equations, we have $n_{c'}(\hat{a}) = n_{c'}(\hat{b})$ and $n_{c'}(\hat{a}) = n_{c'}(\hat{b})$. Together with $n_{c'}(\hat{a}) + n_{c'}(\hat{b}) = x_6 n_{c'}(a) + n_{c'}(b) = x_6 n_{c'}(a)$, is a direct consequence.

These equations give us the tools to analyze the given schedule with makespan $W$. To this point we have proved that we can assume that the machines $M_1$ to $M_4$ contain the correct sets of jobs. Consider the jobs from the sets $A$, $B$, $a$, $b$, and $c$. Note that if one job from the set $A \cup B \cup a \cup b \cup c$ is started no (other) job from the set $A \cup B \cup c$ can be processed at the same time, since these jobs will always share at least one machine when processed. The next step is to prove that the jobs in these sets appear in the correct order, namely $B_0, c_0, A_0, (b_1), B_1, c_1, A_1, \ldots$ see Fig. $5$. We will prove this claim in two steps. First, we will show that in this schedule the first and last jobs have to be elements from the set $A \cup B$ (see Lemma 3). Afterward, we will prove that between two successive jobs from the set $A$ there appears exactly one jobs from each set $B$, $a$, $b$, and $c$, and prove an analogue statement for two successive jobs from the set $B$ (see Lemma 4).

With the knowledge gathered in the proofs of Lemma 3 and Lemma 4, we can prove that the given schedule can be transformed such that all jobs are scheduled contiguously, i.e., on an interval of machines, and that $I$ has to be a Yes-instance of
3-Partition (see Lemma 5). In the following, we write \(=_{(l)}\), when the equation \((l)\) is the reason why the equality is true.

**Lemma 3** One job \(i \in A \cup B\) is the first job which is processed, i.e., \(\sigma(i) = 0\). Furthermore one job from the set \(j \in A \cup B\) is the last job to be processed in the schedule, i.e., \(\sigma(i) + p(j) = W\).

**Proof** Let \(i := \arg \min_{i \in A \cup B} \sigma(i)\) be the job with the smallest start point in \(A \cup B\), (i.e., \(n_i(A) = 0 = n_i(B)\)). If \(i \in A\) it holds that
\[
0 = n_i(B) = (1) n_i(\alpha) + n_i(\{\lambda_1\}) = (1) n_i(a) + n_i(\{\lambda_1\})
\]
and therefore \(n_i(a) = n_i(\alpha) = 0 = n_i(\{\lambda_1\})\). The jobs \(a \cup \alpha \cup \{\lambda_1\} \cup A\) are the only jobs, which are contained on machine \(M_1\). Since \(n_i(A) = 0\) as well, it has to be that \(\sigma(i) = 0\). If \(i \in B\) we can prove \(\sigma(i) = 0\) analogously using equality (2).

Since the schedule stays valid if we mirror the schedule such that the new start points are \(s'(i) = W - \sigma(i) - p(i)\) for each job \(i\), the last job has to be in the set \(A \cup B\) as well. □

Next, we will show that the items in the sets \(A\) and \(B\) have to be scheduled alternately. Let \((A_0, \ldots, A_z)\) be the set \(A\) and \((B_0, \ldots, B_z)\) be the set \(B\) each ordered by increasing size of the starting points. Simply swap the jobs if they do not have this order.

**Lemma 4** If \(\sigma(B_0) = 0\), it holds for each item \(i \in \{0, \ldots, z\}\) that
\[
i = n_{A_i}(A) = n_{A_i}(B) - 1 = n_{A_i}(c) - 1 = n_{A_i}(\alpha) = n_{A_i}(b) = n_{A_i}(a),
\]
and
\[
i = n_{B_i}(B) = n_{B_i}(A) = n_{B_i}(\beta) = n_{B_i}(c) = n_{B_i}(a) = n_{B_i}(b),
\]
and \(n_{A_i}(\{\lambda_1\}) = 1\) as well as \(n_{B_i}(\{\lambda_2\}) = 0\).

**Proof** We will prove this lemma by proving the following claim:

**Claim 1** If \(\sigma(B_0) = 0\), it holds for each item \(i \in \{0, \ldots, z\}\) that
\[
n_{A_i}(A) = n_{A_i}(B) - n_{A_i}(\{\lambda_1\})
\]
and \(n_{A_i}(\{\lambda_1\}) = 1\).

We will prove this claim inductively and per contradiction. Assume for contradiction that
\[
n_{A_0}(B) - n_{A_0}(\{\lambda_1\}) > n_{A_0}(A) = 0.
\]
Therefore, we have \(1 \leq n_{A_0}(B) - n_{A_0}(\{\lambda_1\})\). Let \(a' \in a, b' \in b\) and \(c' \in c\) be the first started jobs in their sets. Since
\[
n_{A_0}(b) = (1) n_{A_0}(a) = (1) n_{A_0}(c) - n_{A_0}(\{\lambda_1\}) = (1) n_{A_0}(B) - n_{A_0}(\{\lambda_1\}) \geq 1,
\]
the jobs \(a', b'\) and \(c'\) start before \(A_0\). It holds that \(n_{B'}(c) = (4) n_{B'}(A) = 0\). Therefore, \(c'\) has to start after \(b'\) resulting in \(n_{c'}(b) \geq 1\). Furthermore, we have...
\[ n_{a'}(c) = (3) \ n_{a'}(B) \geq 1. \] Hence, \( c' \) has to start before \( a' \) resulting in \( n_{c'}(a) = 0. \) In total we have \( 1 \leq n_{c'}(b) = (5) \ n_{c'}(a) = 0, \) a contradiction. Therefore, we have \[ n_{A_0}(B) - n_{A_0}([\lambda_1]) \leq n_{A_0}(A) = 0. \] As a consequence, it holds that \( 1 \leq n_{A_0}(B) \leq n_{A_0}([\lambda_1]) \leq 1 \) and we can conclude \( n_{A_0}(B) = 1 = n_{A_0}([\lambda_1]) \) as well as \( n_{A_0}(B) - n_{A_0}([\lambda_1]) = n_{A_0}(A). \) Therefore \( 1 = n_{A_1}([\lambda_1]) \) holds for all \( i \in \{0, \ldots, z\}. \)

To this point we have proved the induction basis.

For the induction step it is enough to prove that \( n_{A_i}(B) = 1 = n_{A_i}(A) \) for all \( i \in \{0, \ldots, z\}. \) To prove this, we first show that if this condition holds for all \( i' \) up to an \( i \in \{0, \ldots, z\} \) we can derive the (6) and an equation which is similar to (7) for all these \( i'. \) Choose \( i \in \{0, \ldots, z\} \) such that

\[ n_{A_i'}(B) - 1 = n_{A_i'}(A) \quad (8) \]

for all \( i' \in \{0, \ldots, i\}. \) A direct consequence from this equation is the (6) for all \( i' \in \{0, \ldots, i\}: \)

\[ i' = n_{A_i'}(A) = (8) n_{A_i'}(B) - 1 = (1) n_{A_i'}(c) - 1 = (1) n_{A_i'}(\alpha) = (1) n_{A_i'}(b) = (1) n_{A_i'}(a). \]

Furthermore, we have \( n_{B_i}(B) = i = n_{A_i}(A) = n_{A_i}(B) - 1. \) Therefore \( B_i \) has to be scheduled before \( A_i. \) Additionally, we have \( n_{B_i}(B) - 1 = n_{B_{i-1}}(B) = i - 1 = n_{A_{i-1}}(A) = n_{A_{i-1}}(B) - 1, \) so \( B_i \) has to be scheduled after \( A_{i-1}. \) Therefore, we have \( n_{B_i}(B) = n_{B_i}(A) \) and the following equation is a consequence for all \( i' \in \{0, \ldots, i\}: \)

\[ i' = n_{B_i'}(B) = n_{B_i'}(A) = (2) n_{B_i'}(c) = (2) n_{B_i'}(\beta) + n_{B_i'}([\lambda_2]) \]

\[ = (2) n_{B_i'}(\alpha) + n_{B_i'}([\lambda_2]) = (2) n_{B_i'}(b) + n_{B_i'}([\lambda_2]). \quad (9) \]

We still have to prove Claim 1, to prove the lemma. To this point, we have proved the base of the induction. However, we still have to prove \( n_{A_{i+1}}(B) - 1 = n_{A_{i+1}}(A) \) since we already know that \( n_{A_i}([\lambda_1]) = 1 \) for all \( i \in \{0, \ldots, z\}. \) We will prove this in two steps:

**Claim 2** \( n_{A_{i+1}}(B) - 1 \leq n_{A_{i+1}}(A) \)

**Assume** for contradiction that \( n_{A_{i+1}}(B) - 1 > n_{A_{i+1}}(A). \) As a consequence, we have

\[ 1 = n_{A_{i+1}}(A) - n_{A_i}(A) < n_{A_{i+1}}(B) - 1 = (n_{A_i}(B) - 1) = n_{A_{i+1}}(B) - n_{A_i}(B), \]

and hence, \( n_{A_{i+1}}(B) - n_{A_i}(B) \geq 2. \) Therefore, there are jobs \( B_{i+1}, B_{i+2} \in B \) that are scheduled between \( A_i \) and \( A_{i+1}. \) Since we have

\[ 2 \leq n_{A_i}(B) - 1 = (n_{A_i}(B) - 1) = (1) n_{A_i}(c) - 1 = (n_{A_{i+1}}(c) - 1) \]

\[ = (1) n_{A_i}(b) - n_{A_{i+1}}(b) = (1) n_{A_i}(a) - n_{A_{i+1}}(a) \]

there have to be two jobs \( a', a'' \in a, b', b'' \in b \) and \( c', c'' \in c \) that are scheduled between \( A_i \) and \( A_{i+1} \) as well. W.l.o.g we assume that \( \sigma(a') \leq \sigma(a''), \sigma(b') \leq \sigma(b'') \) and \( \sigma(c') \leq \sigma(c''). \)

Next, we will deduce in which order the jobs \( a', a'', b', b'', c', c'', B_{i+1}, \) and \( B_{i+2} \) appear in the schedule. It holds that

\[ n_{b''}(c) = (4) n_{b''}(A) \quad \sigma(a_i) < \sigma(b'') \]

\[ = n_{A_i}(A) + 1 = (8) n_{A_i}(B) = (1) n_{A_i}(c) \]

Springer
since \( b'' \) starts after \( A_i \) but before \( A_{i+1} \). Therefore, \( b' \) and \( b'' \) have to finish before \( c' \), i.e., \( \sigma(b') < \sigma(b'') < \sigma(c') \), since no job from the set \( c \) can be scheduled between \( A_i \) and \( b'' \). As a consequence, we have

\[
n_{c'}(a) = (5) n_{c'}(b) \quad \sigma(A_i) < \sigma(b') < \sigma(b'') < \sigma(c') \geq n_{A_i}(b) + 2 = (1) n_{A_i}(a) + 2.
\]

Hence, \( a'' \) has to start before \( c' \) as well. Additionally, it holds that

\[
n_{B_{i+2}}(c) = (2) n_{B_{i+2}}(A) \quad \sigma(A_i) < \sigma(B_{i+2}) \geq n_{A_i}(A) + 1 = (8) n_{A_i}(B) = (1) n_{A_i}(c).
\]

since \( B_{i+2} \) starts after \( A_i \) but before \( A_{i+1} \). As a consequence, \( B_{i+2} \) has to start before \( c' \). Additionally, it holds that

\[
n_{a''}(B) = (3) n_{a''}(c) \quad \sigma(A_i) < \sigma(a'') < \sigma(c') = (1) n_{A_i}(B).
\]

Therefore, \( a'' \) has to start before \( B_{i+1} \), since there can not be a job from the set \( B \) scheduled between \( A_i \) and \( a'' \).

To this point, we have deduced that the jobs have to appear in the following order in the schedule: \( A_i, a', a'', B_{i+1}, B_{i+2}, c', c'', A_{i+1} \). However, this schedule is not feasible, since we have

\[
n_{A_i}(a) + 2 \quad \sigma(A_i) < \sigma(a') < \sigma(a'') < \sigma(B_{i+1}) \leq n_{B_{i+1}}(a) \leq (2) n_{B_{i+1}}(A) \quad \sigma(A_i) < \sigma(B_{i+1}) < \sigma(A_{i+1}) \leq n_{A_i}(A) + 1 = (1) n_{A_i}(a) + 1,
\]

a contradiction. 

Therefore, the assumption \( n_{A_i+1}(B) - 1 > n_{A_i+1}(A) \) was wrong, and it holds that \( n_{A_i+1}(B) - 1 \leq n_{A_i+1}(A) \) proving Claim 2.

**Claim 3** \( n_{A_{i+1}}(B) - 1 \geq n_{A_{i+1}}(A) \)

Assume for contradiction that \( n_{A_i+1}(B) - 1 < n_{A_i+1}(A) \). As a consequence, it holds that \( n_{A_i+1}(B) = n_{A_i}(B) \) since

\[
n_{A_i}(B) - 1 \leq n_{A_i+1}(B) - 1 \quad \text{assumption} \leq n_{A_{i+1}}(A) - 1 = n_{A_i}(A) = n_{A_i}(B) - 1.
\]

Furthermore, there has to be at least one job \( B_{i+1} \in B \) that starts after \( A_{i+1} \) since \( |A| = |B| \). Therefore, we have \( n_{B_{i+1}}(c) - n_{B_i}(c) = n_{B_{i+1}}(A) - n_{B_i}(A) \geq 2 \). As a consequence, there are jobs \( c', c'' \in c \) which are scheduled between \( B_i \) and \( B_{i+1} \). Let \( c' \) be the first job in \( c \) scheduled after \( B_i \) and \( c'' \) be the next. Since we do not know the value of \( n_{B_i}([\lambda_2]) \) or \( n_{B_{i+1}}([\lambda_2]) \), we can just deduce from (2) that \( n_{B_{i+1}}(a) - n_{B_i}(a) \geq 1 \). Therefore, there has to be a job \( a' \in a \) that is scheduled between \( B_i \) and \( B_{i+1} \).
We will now look at the order in which the jobs $A_i$, $A_{i+1}$, $c'$, $c''$ and $a'$ have to be scheduled. First, we know that $A_i$ and $A_{i+1}$ have to be scheduled between $c'$ and $c''$, since
\[ n_{A_i}(c) = (1) n_{A_i}(B) \stackrel{\sigma(B_i)<\sigma(A_i)<\sigma(B_{i+1})}{=} n_{B_i}(B) + 1 = (9) n_{B_i}(A) + 1 = (2) n_{B_i}(c) + 1 \]
and
\[ n_{A_{i+1}}(c) = (1) n_{A_{i+1}}(B) \stackrel{\sigma(B_i)<\sigma(A_{i+1})<\sigma(B_{i+1})}{=} n_{B_i}(B) + 1 = (9) n_{B_i}(A) + 1 = (2) n_{B_i}(c) + 1, \]
and hence there has to be exactly one job from the set $c$ scheduled between $B_i$ and $A_i$, as well as $B_i$ and $A_{i+1}$. Furthermore, we know that $a'$ has to be scheduled between $c'$ and $c''$ as well, since
\[ n_{a'}(c) = (3) n_{a'}(B) \stackrel{\sigma(B_i)<\sigma(a')<\sigma(c')}{=} n_{B_i}(B) + 1 = (9) n_{B_i}(A) + 1 = (2) n_{B_i}(c) + 1. \]
As a consequence, we can deduce that there is a job $b' \in b$ which is scheduled between $c'$ and $c''$, since
\[ n_{c'}(b) = (5) n_{c'}(a) \stackrel{\sigma(c')<\sigma(a')<\sigma(c'')}{=} n_{c'}(a) + 1 = (5) n_{c'}(b) + 1. \]
We know about this $b'$ that
\[ n_{b'}(A) = (4) n_{b'}(c) \stackrel{\sigma(B_i)<\sigma(c')<\sigma(b')<\sigma(c'')}{=} n_{B_i}(c) + 1 = (2) n_{B_i}(A) + 1, \]
so $b'$ has to be scheduled between $A_i$ and $A_{i+1}$.

In summary, the jobs are scheduled as follows: $B_i$, $c'$, $A_i$, $b'$, $A_{i+1}$, $c''$, $B_{i+1}$. However, this schedule is infeasible since
\[ n_{A_i}(b) = (1) n_{A_i}(B) - 1 \stackrel{\text{assumption}}{=} n_{A_{i+1}}(B) - 1 = (1) n_{A_{i+1}}(b) = n_{A_i}(b) + 1, \]
a contradiction. 

As a consequence, it has to hold that $n_{A_{i+1}}(B) - 1 \geq n_{A_i}(A)$ proving Claim 3. Altogether as a consequence of Claim 2 and Claim 3, we have proved that $n_{A_{i+1}}(B) - n_{A_i+1}([\lambda_1]) = n_{A_i+1}(A)$ for all $i \in \{0, \ldots, z\}$. This concludes the proof of the Claim 1.

Last we have to prove the (7). To do this we have to prove that $n_{B_i}([\lambda_1]) = 0$ for all $i \in \{0, \ldots, z\}$. We do this by proving the following claim.

**Claim 4** $\lambda_2$ is scheduled after the last job in $B$.

To prove the (7), we have to prove that $n_{B_i}([\lambda_2]) = 0$ for each $i \in \{0, \ldots, z\}$, i.e., we have to prove Claim 4. Assume there is an $i \in \{0, \ldots, z\}$ with $n_{B_i}([\lambda_2]) > 0$. Let $i$ be the smallest of these indices. We know that
\[ i - 1 = (9) n_{B_i}(A) - 1 = n_{B_i}(A) - n_{B_i}([\lambda_2]) = (2) n_{B_i}(a). \]
Since $n_{A_i}(b) = (1) n_{A_i}(a) = (6) i = n_{B_i}(a) + 1 = (2) n_{B_i}(b) + 1$ there has to be an unique $a' \in a$ and an unique $b' \in b$ scheduled between $B_i$ and $A_i$. Furthermore, since $n_{A_i}(c) = (6) i + 1$ and $n_{B_i}(c) = (9) i$, there has to be a $c' \in c$ scheduled
between $B_i$ and $A_i$ as well. At the start of $b'$ it holds that $n_{B'}(c) = (4) n_{B'}(A) = n_{A_{i-1}}(A) + 1 = (4) n_{A_{i-1}}(c)$, so $b'$ has to start before $c'$. Additionally, at the start of $a'$ we have $n_{a'}(c) = (4) n_{a'}(B) = n_{B_i}(B) + 1 = (9) n_{B_i}(c) + 1$ and therefore $a'$ has to start after $c'$. In total, the jobs appear in the following order: $B_i, b', c', a', A_i$. But this can not be the case, since we have

$$n_{B_i-1}(a) \left(\sigma(c') < \sigma(a')\right) = n_{c'}(a) = (5) n_{c'}(b) \left(\sigma(B_i) < \sigma(b') < \sigma(c')\right) n_{B_i-1}(b) + 1 = (2) n_{B_i-1}(a) + 1.$$

Hence, we have contradicted that assumption. Hence, we have $n_{B_i}(\{\lambda_2\}) = 0$ for all $i \in \{0, \ldots, z\}$ and (7) is a consequence:

$$n_{B_i}(b) = n_{B_i}(a) = n_{B_i}(c) = n_{B_i}(\beta) = n_{B_i}(A) = n_{B_i}(B) = i.$$

A direct consequence of Lemma 4 is that the last job on $M_2$ is a job in $A$. Since the (1) and (2), as well as (3) and (4), are symmetric, we can deduce an analogue statement if the first job on $M_2$ is in $A$. More precisely, we can show that $n_{B_i}(A) - n_{B_i}(\{\lambda_2\}) = n_{B_i}(B)$ and $n_{B_i}(\{\lambda_2\}) = 1$ for each $B_i \in B$ in this case. This would imply that the last job on $M_2$ is a job in $B$. Since we can mirror the schedule such that the last job is the first job, we can suppose that the first job on $M_2$ is a job in $B$.

At this point, we know which machines process which jobs and for all the jobs using more than one machine we know the rough order of their processing by the (6) and (7). These are all the tools we need to prove that if the optimal schedule for the scheduling instance derived from $I$ has makespan $W$ then $I$ is a Yes-instance for 3-Partition and we will prove it in the next lemma.

**Lemma 5** If the optimal schedule for the scheduling instance derived from $I$ has makespan $W$ then $I$ is a Yes-instance for 3-Partition and we can transform the schedule such that all jobs are scheduled on contiguous machines.

**Proof** First, we will prove that $M_1$ processes the jobs $A \cup a \cup \alpha \cup \{\lambda_1\}$ in the order $\lambda_1, A_0, a_1, a_1, A_1, a_2, a_2, A_2, \ldots, a_z, a_z, A_z$, where $a_i \in a$ and $\alpha_i \in \alpha$ for each $i \in \{1, \ldots, z\}$, see Fig. 6. Lemma 4 ensures that the first job on $M_1$ is the job $\lambda_1$. Furthermore, since $0 = n_{A_0}(A) = (6) n_{A_0}(a) = (6) n_{A_0}(\alpha)$, the second job on $M_1$ is $A_0$. For each $i \in \{1, \ldots, z\}$ it holds that $n_{A_i}(\alpha) = (6) n_{A_i-1}(\alpha) + 1$ and $n_{A_i}(a) = (6) n_{A_i-1}(a) + 1$. Therefore, there is exactly one job $a_i \in a$ and one job $\alpha_i \in \alpha$ scheduled between the jobs $A_{i-1}$ and $A_i$. It holds that $n_{A_i-1}(a) + 1 = (6) i = (7) n_{B_i}(a)$. Therefore, $a_i$ has to be scheduled between $A_{i-1}$ and $B_i$. As a consequence, we have $n_{a_i}(\alpha) + 1 = n_{a_i}(a) + n_{a_i}(\{\lambda_1\}) = (3) n_{a_i}(B) = n_{B_i}(B) = (7) n_{B_i}(a) = n_{B_i}(a) + 1$.

![Fig. 6](image-url) Proved positions of the jobs in the sets $A$, $a$, and $\alpha$

\[\begin{array}{c|cccccccccc}
M_4 & B_4 & c_0 & B_3 & c_1 & B_2 & c_1 & B_3 & c_1 & B_4 & \lambda_2 \\
M_3 & B_3 & c_1 & B_2 & c_1 & B_3 & c_1 & B_4 & c_1 & \lambda_2 \\
M_2 & B_2 & c_1 & B_3 & c_1 & B_4 & c_1 & \lambda_2 \\
M_1 & B_1 & c_1 & \lambda_1 \\
\end{array}\]
Therefore, \( a_i \) has to be scheduled before \( \alpha_i \) and the jobs appear in machine \( M_1 \) in the described order. As a result, we know about the start point of \( A_i \) that
\[
\sigma(A_i) = p(\lambda_i) + i_p + i_p + i_p = (i + 1)(D^2 + D^3) + i(D^4 + D^5 + D^6 + D^7) + (7z + 1)D^8.
\]

Now, we will show that the machine \( M_4 \) processes the jobs \( B \cup B \cup \beta \cup \lambda_2 \) in the order \( B_0, b_1, B_1, b_2, B_2, \ldots, b_z, B_z, \lambda_2 \) see Fig. 7. The first job on \( M_4 \) is the job \( B_0 \), since Lemma 3 states that one of the jobs in \( A \cup B \cup \beta \cup \lambda \) has start point 0 and we decided w.l.o.g. that \( B_0 \) is this job. Equation (7) ensures that between the jobs \( B_i \) and \( B_{i+1} \) there is scheduled exactly one job \( b_{i+1} \) and exactly one job \( \beta_{i+1} \). It holds that \( n_{A_i}(b) + 1 = (6) i + 1 = (7) n_{B_{i+1}}(b) \). Therefore, \( b_{i+1} \) has to be scheduled between \( A_i \) and \( B_{i+1} \). As a consequence, it holds that
\[
n_{b_{i+1}}(\beta) = n_{b_{i+1}}(\beta) + n_{b_{i+1}}(\lambda_2) = (4) n_{b_{i+1}}(A) = (3) n_{b_{i+1}}(A) = (7) n_{B_{i+1}}(b) + 1.
\]

Hence, \( b_{i+1} \) has to be scheduled after \( \beta_{i+1} \) and the jobs on machine \( M_4 \) appear in the described order. As a result, we know about the start point of \( B_i \) that
\[
\sigma(B_i) = i + i_p + i_p + i_p = iD^2 + iD^3 + iD^4 + iD^5 + iD^6 + iD^7 + (i(7z - 1))D^8.
\]

Next, we can deduce that the jobs in \( c \) are scheduled as shown in Fig. 7. We have \( n_{B_i}(c) = (7) i = (6) n_{A_i}(c) - 1 \). Therefore, there exists an \( c' \in c \) for each \( i \in \{0, \ldots, z\} \), which is scheduled between \( B_i \) and \( A_i \). The processing time between \( B_i \) and \( A_i \) is exactly \( \sigma(A_i) - \sigma(B_i) = D^3 + (z + i)D^8 \). As a consequence, one can see with an inductive argument that \( c_i \) in \( c \) with \( p(c_i) = D^3 + (z + i)D^8 \) has to be positioned between \( B_i \) and \( A_i \), since the job in \( c \) with the largest processing time, \( c_z \), fits between \( B_z \) and \( A_z \) only.

In this step, we will transform the schedule such that all jobs are scheduled on contiguous machines. To this point, this property is obviously fulfilled by the jobs in \( A \cup B \cup c \). However, the jobs in \( a \cup b \) might be scheduled on non-contiguous machines. We know that the \( a_i \) and \( b_i \) are scheduled between \( A_{i-1} \) and \( B_i \). One part of \( a_i \) is scheduled on \( M_1 \) and one part of \( b_i \) is scheduled on \( M_4 \), while each other part is scheduled either on \( M_2 \) or on \( M_3 \) but both parts on different machines, because \( \sigma(B_i) - \sigma(A_{i-1}) - p(A_i) = D^5 + D^6 + D^7 + (6z - i)D^8 < D^5 + 2D^6 + D^7 + 6zD^8 = p(a_i) + p(b_i) \) for each \( i \in \{0, \ldots, z\} \). Since \( A_i \) and \( B_{i+1} \) both are scheduled on machines \( M_2 \) and \( M_3 \), we can swap the content of the machines between these jobs.

![Diagram](https://example.com/diagram.png)

**Fig. 7** Proved positions of the jobs in the sets \( B, b, \) and \( \beta \)

© Springer
such that the other part of $a_i$ is scheduled on $M_2$ and the other part of $b_i$ is scheduled on $M_3$. We do this swapping step for all $i \in \{0, \ldots, z-1\}$ such that all other parts of jobs in $a$ are scheduled on $M_2$ and all other part of jobs in $b$ are scheduled on $M_3$ respectively. After this swapping step, all jobs are scheduled on contiguous machines.

Now, we will show that $I$ is a Yes-instance. To this point we know that $M_2$ contains the jobs $A \cup B \cup a \cup c$. Since $\tilde{a} = a$ and $|\tilde{a}| = |\tilde{\gamma}|$, it has to hold by Lemma 1 that $\tilde{\gamma} = \gamma$ implying that $M_2$ contains all jobs in $\gamma$. Furthermore, since $\tilde{b} = b$ and $|\tilde{b}| = |\delta|$, we have $\delta = \emptyset$ and therefore $M_2$ does not contain any job in $\delta$. Besides the jobs $A \cup B \cup a \cup c \cup \gamma$, $M_2$ processes further jobs with total processing time $zD$. Therefore, all the jobs in $P$ are processed on $M_2$. We will now analyse where the jobs in $\gamma$ are scheduled. The only possibility where these jobs can be scheduled is the time between the end of $a_i$ and the start of $B_i$ for each $i \in \{1, \ldots, z\}$ since at each other time the machine is occupied by other jobs. The processing time between the end of $a_i$ and the start of $B_i$ is exactly $\sigma(B_i) - \sigma(A_{i-1}) - p(A_{i-1}) - p(a_i) = D^7 + (3z - i)D^8$. The job in $\gamma$ with the largest processing time is the job $\gamma_1$ with $p(\gamma_1) = D^7 + (3z - 1)D^8 - D$. This job only fits between $a_1$ and $B_1$. Inductively we can show that $\gamma_i \in \gamma$ with $p(\gamma_i) = D^7 + (3z - i)D^8 - D$ has to be scheduled between $a_i$ and $B_i$ on $M_2$. Furthermore since $p(\gamma_i) = D^7 + (3z - i)D^8 - D$ and the processing time between the end of $a_i$ and the start of $B_i$ is $D^7 + (3z - i)D^8$, there is exactly $D$ processing time left. This processing time has to be occupied by the jobs in $P$ since this schedule has no idle times. Therefore, we have for each $i \in \{1, \ldots, z\}$ a subset $P_i \subseteq P$ containing jobs with processing times adding up to $D$ such that $P_1 \cup \ldots \cup P_z = P$. As a consequence $I$ is a Yes-instance. 

\section{3 Hardness of Strip Packing}

In this section, we will prove the Theorem 2. This can be done straight forward, by using the reduction from above. Note that in the transformed optimal schedule, all jobs are scheduled on contiguous machines, i.e., the machines the jobs are scheduled on are Neighbors in the natural order ($M_1, M_2, M_3, M_4$). As a consequence, we have proved that this problem is strongly NP-complete even if we restrict the set of feasible solutions to those where all jobs are scheduled on contiguous machines. We will now describe how this insight delivers a lower bound of $\frac{5}{4}$ for the best possible approximation ratio for pseudo-polynomial Strip Packing and in this way prove the Theorem.

To show our hardness result for Strip Packing, let us consider the following instance. We define $W := (z + 1)(D^2 + D^3 + D^4) + z(D^5 + D^6 + D^7) + z(7z + 1)D^8$ as the width of the strip, i.e., it is the same as the considered makespan in the scheduling problem. For each job $j$ defined in the reduction above, we introduce an item $i$ with $w(i) = p(j)$ and height $h(i) = q(j)$. Now, we can show analogously that if the 3-Partition instance is a Yes-instance, there is a packing of height 4 (one example is the packing in Fig. 4); and on the other hand if there is a packing with height 4, the 3-Partition instance has to be a Yes-instance. If the 3-Partition instance is a No-instance, the optimal packing has a height of at least 5 since the optimal height for
this instance is integral. Therefore, we cannot approximate Strip Packing better than \( \frac{5}{4} \) in pseudo-polynomial time unless \( P = NP \).

### 3.1 Variants of Strip Packing

Lastly we look at a variant of the Strip Packing problem called Scheduling of Contiguous Moldable Parallel Tasks. In this problem setting we are given an (arbitrary large) set of \( m \) machines, which have some kind of total order, and a set of parallel tasks \( J \). Each task \( j \in J \) has a set \( D_j \subseteq \{1, \ldots, m\} \) of machine amounts it can be processed on, e.g., if \( D_j = \{3, 7\} \) the job \( j \) can be processed either on three or on seven machines. For each \( q \in D_j \) it has an individual processing time \( p(j, q) \in \mathbb{N}_{>0} \cup \{\infty\} \). A schedule \( S \) is given by two functions \( \sigma : J \to \mathbb{N} \) and \( \rho : J \to 2^{\{1,\ldots,m\}} \). The function \( \sigma \) maps each job to a start point in the schedule, while \( \rho \) maps each job to an interval of machines it is processed on. A schedule is feasible if each machine processes at most one job at a time and each job is processed on the required number of machines, (i.e., \( |\rho(j)| \in D_j \)).

This problem directly contains the Strip Packing problem as a special case by setting \( D_i = \{w(i)\} \) and \( p(i, w(i)) = h(i) \) for each item \( i \in I \), and hence, it is NP-hard to approximate it better than \( \frac{5}{4} \) in pseudo-polynomial time. However, it is also NP-hard to find a better approximation than \( \frac{5}{4} \) if we require \( D_j = \{1, \ldots, m\} \) and \( p(j, q) \in \mathbb{N}_{>0} \) for each job and number of machines. To show this, we use the reduction from above. The number of machines is \( m := (z + 1)(D^2 + D^3 + D^4) + z(D^5 + D^6 + D^7) + z(7z + 1)D^8 \). For each item \( i \in I \) constructed for the Strip Packing instance, we introduce one job \( i \) with \( p(i, q) = 5 \), if \( q < w(i) \) and \( p(i, q) = h(i) \), if \( q \geq w(i) \). Obviously a schedule with height 4 can be found if and only if each job \( i \) uses \( w(i) \) machines and the 3-Partition instance is a Yes-instance. Otherwise the optimal schedule has a makespan of 5. Hence it is not possible to find an algorithm with approximation ratio better than \( 5/4 \), unless \( P = NP \).

Note that in this construction the processing time function is not monotone, i.e., we do not have \( p(j, q) \cdot q \leq p(j, q + 1) \cdot (q + 1) \) for each \( q \in \{1, \ldots, m - 1\} \). Hence, there could be a PTAS for the monotone case.

### 4 Conclusion

In this paper, we positively answered the long standing open question whether the problem \( P4|\text{size}_j|C_{\text{max}} \) is strongly NP-complete. This closes the gap between strongly NP-completeness for at least 5 machines, and the possibility to solve the problem in pseudo-polynomial time for at most 3 machines.

Furthermore, we have improved the lower bound for pseudo-polynomial Strip Packing to \( \frac{5}{4} \). The best known published algorithm has an approximation ratio of \( \frac{4}{3} \). This leaves a gap between the lower bound and the best known algorithm. However, very recently we were able to find a pseudo-polynomial time algorithm with approximation ratio \( \frac{5}{4} + \varepsilon \) \([20]\), which closes this gap.

Lastly, we have considered Scheduling of Contiguous Moldable Parallel Tasks and proved that in the non-monotone case there is no pseudo-polynomial time algorithm...
with approximation ratio better than 5/4 unless $P = NP$. However, in the monotone case, finding a PTAS might be possible. In our opinion, it is an interesting open problem, whether there is a PTAS for the monotone case or not, especially since there is an FPTAS for the case that $m > 8n/\varepsilon$, see [18].

Acknowledgements The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work was supported by German Research Foundation (DFG) project JA 612 /20-1.

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Adamaszek, A., Kociumaka, T., Pilipczuk, M., Pilipczuk, M.: Hardness of approximation for strip packing. TOCT 9(3), 14:1–14:7 (2017). https://doi.org/10.1145/3092026
2. Amoura, A.K., Bampis, E., Kenyon, C., Manoussakis, Y.: Scheduling independent multiprocessor tasks. Algorithmica 32(2), 247–261 (2002). https://doi.org/10.1007/s00453-001-0076-9
3. Baker, B.S., Brown, D.J., Howard, P.K.: 5/4 algorithm for two-dimensional packing. J. Algor. 2(4), 348–368 (1981). https://doi.org/10.1016/0196-6774(81)90034-1
4. Baker, B.S., Coffman, E.G. Jr., Rivest, R.L.: Orthogonal packings in two dimensions. SIAM J. Comput. 9(4), 846–855 (1980). https://doi.org/10.1137/0209064
5. Bansal, N., Correa, J.R., Kenyon, C., Sviridenko, M.: Bin packing in multiple dimensions Inapproximability results and approximation schemes. Math. Oper. Res. 31(1), 31–49 (2006). https://doi.org/10.1287/moor.1050.0168
6. Bougeret, M., Dutot, P.-F., Jansen, K., Robenek, C., Trystram, D.: Approximation algorithms for multiple strip packing and scheduling parallel jobs in platforms. Discret. Math. Algor. Appl. 3(4), 553–586 (2011). https://doi.org/10.1142/S1793830911001413
7. Christensen, H.I., Khan, A., Pokutta, S., Tetali, P.: Approximation and online algorithms for multi-dimensional bin packing: A survey. Computer Science Review. https://doi.org/10.1016/j.cosrev.2016.12.001 (2017)
8. Coffman, E.G. Jr., Garey, M.R., Johnson, D.S., Tarjan, R.E.: Performance bounds for level-oriented two-dimensional packing algorithms. SIAM J. Comput. 9(4), 808–826 (1980). https://doi.org/10.1137/0209062
9. Jianzhong, D., Leung, J.Y.-T.: Complexity of scheduling parallel task systems. SIAM J. Discret. Math. 2(4), 473–487 (1989). https://doi.org/10.1137/0402042
10. Feldmann, A., Sgall, J., Teng, S.-H.: Dynamic scheduling on parallel machines. Theor. Comput. Sci. 130(1), 49–72 (1994). https://doi.org/10.1016/0304-3975(94)90152-X
11. Gálvez, W., Grandoni, F., Ingala, S., Khan, A.: Improved pseudo-polynomial-time approximation for strip packing. In: 36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), pp. 9:1–9:14. https://doi.org/10.4230/LIPIcs.FSTTCS.2016.9 (2016)
12. Garey, M.R., Graham, R.L.: Bounds for multiprocessor scheduling with resource constraints. SIAM J. Comput. 4(2), 187–200 (1975). https://doi.org/10.1137/0204015
13. Garey, M.R., Johnson, D.S.: Computers and Intractability: A guide to the theory of NP-completeness (1979)
14. Golan, I.: Performance bounds for orthogonal oriented two-dimensional packing algorithms. SIAM J. Comput. 10(3), 571–582 (1981). https://doi.org/10.1137/0210042
15. Harren, R., Jansen, K., Prádel, L., Rob van, S.: A $(5/3 + \varepsilon)$-approximation for strip packing. Comput. Geom. 47(2), 248–267 (2014). https://doi.org/10.1016/j.comgeo.2013.08.008
16. Harren, R., van Stee, R.: Improved absolute approximation ratios for two-dimensional packing problems. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and
17. Jansen, K.: A \((3/2+\varepsilon)\) approximation algorithm for scheduling moldable and non-moldable parallel tasks. In: 24th ACM Symposium on Parallelism in Algorithms and Architectures, (SPAA), pp. 224–235. https://doi.org/10.1145/2312005.2312048 (2012)

18. Jansen, K., Land, F.: Scheduling monotone moldable jobs in linear time. In: 2018 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2018, Vancouver, BC, Canada, May 21-25, 2018, pp. 172–181. https://doi.org/10.1109/IPDPS.2018.00027 (2018)

19. Jansen, K., Porkolab, L.: Linear-time approximation schemes for scheduling malleable parallel tasks. Algorithmica 32(3), 507–520 (2002). https://doi.org/10.1007/s00453-001-0085-8

20. Jansen, K., Rau, M.: Closing the gap for pseudo-polynomial strip packing. arXiv:1712.04922 (2017)

21. Jansen, K., Rau, M.: Improved approximation for two dimensional strip packing with polynomial bounded width. In: WALCOM: Algorithms and Computation, Volume 10167 of LNCS, pp. 409–420. https://doi.org/10.1007/978-3-319-53925-6_32 (2017)

22. Jansen, K., Solis-Oba, R.: Rectangle packing with one-dimensional resource augmentation. Discret. Optim. 6(3), 310–323 (2009). https://doi.org/10.1016/j.disopt.2009.04.001

23. Jansen, K., Thölle, R.: Approximation algorithms for scheduling parallel jobs. SIAM J. Comput. 39(8), 3571–3615 (2010). https://doi.org/10.1137/080753491

24. Kenyon, C., Remila, E.: A near-optimal solution to a two-dimensional cutting stock problem. Math. Oper. Res. 25(4), 645–656 (2000). https://doi.org/10.1287/moor.25.4.645.12118

25. Ludwig, W., Tiwari, P.: Scheduling malleable and nonmalleable parallel tasks. In: 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 167–176 (1994)

26. Nadiradze, G., Wiese, A.: On approximating strip packing with a better ratio than 3/2. In: 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1491–1510. https://doi.org/10.1137/1.9781611974331.ch102 (2016)

27. Schiermeyer, I.: Reverse-fit: A 2-optimal algorithm for packing rectangles. In: 2nd Annual European Symposium on Algorithms (ESA) - Algorithms, pp. 290–299. https://doi.org/10.1007/978-3-540-44946-4_29 (1994)

28. Sleator, D.D.: A 2.5 times optimal algorithm for packing in two dimensions. Inf. Process. Lett. 10(1), 37–40 (1980). https://doi.org/10.1016/0020-0190(80)90121-0

29. Steinberg, A.: A strip-packing algorithm with absolute performance bound 2. SIAM J. Comput. 26(2), 401–409 (1997). https://doi.org/10.1137/S0097539793255801

30. Sviridenko, M.: A note on the kenyon-remila strip-packing algorithm. Inf. Process. Lett. 112(1-2), 10–12 (2012). https://doi.org/10.1016/j.ipl.2011.10.003

31. Turek, J., Wolf, J.L., Philip, S.: Approximate algorithms scheduling parallelizable tasks. In: 4th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp. 323–332. https://doi.org/10.1145/140901.141909 (1992)