Closing the Gap for Single Resource Constraint Scheduling

Klaus Jansen
Universität Kiel, Germany

Malin Rau
Universität Hamburg, Germany

Abstract

In the problem called single resource constraint scheduling, we are given m identical machines and a set of jobs, each needing one machine to be processed as well as a share of a limited renewable resource R. A schedule of these jobs is feasible if, at each point in the schedule, the number of machines and resources required by jobs processed at this time is not exceeded. It is NP-hard to approximate this problem with a ratio better than $3/2$. On the other hand, the best algorithm so far has an absolute approximation ratio of $2 + \varepsilon$. In this paper, we present an algorithm with absolute approximation ratio $(3/2 + \varepsilon)$, which closes the gap between inapproximability and best algorithm with exception of a negligible small ε.

2012 ACM Subject Classification Theory of computation → Packing and covering problems; Theory of computation → Scheduling algorithms

Keywords and phrases resource constraint scheduling, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.53

Related Version *Full Version*: `http://arxiv.org/abs/2107.01613`

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

1 Introduction

In the single resource constraint scheduling problem, we are given m identical machines, a discrete renewable resource with a fixed size $R \in \mathbb{N}$ and a set of n jobs $\mathcal{J}$. Each job has a processing time $p(j) \in \mathbb{Q}$. We define the total processing time of a set of jobs $\mathcal{J}' \subseteq \mathcal{J}$ as $p(\mathcal{J}') := \sum_{j\in\mathcal{J}'} p(j)$. To be scheduled, each job $j \in \mathcal{J}$ needs one of the machines as well as a fix amount $r(j) \in \mathbb{N}$ of the resource which it will allocate during the complete processing time $p(j)$ and which it deallocates as soon as it has finished its processing. Neither machine nor any part of the resource can be allocated by two different jobs at the same time. We define the area of a job as $\text{area}(j) := r(j) \cdot p(j)$ and the area of a set of jobs $\mathcal{J}' \subseteq \mathcal{J}$ as $\text{area}(\mathcal{J}') := \sum_{j\in\mathcal{J}'} r(j) \cdot p(j)$.

A schedule $\sigma : \mathcal{J} \to \mathbb{N}$ maps each job $j \in \mathcal{J}$ to a starting point $\sigma(j) \in \mathbb{Q}$. We say a schedule is feasible if

$$\forall t \in \mathbb{Q} : \sum_{j:t\in[\sigma(j),\sigma(j)+p(j))} r(j) \le R \text{ and} \qquad \textit{(resource condition)}$$

$$\forall t \in \mathbb{Q} : \sum_{j:t\in[\sigma(j),\sigma(j)+p(j))} 1 \le m. \qquad \textit{(machine condition)}$$

Given a schedule σ where these two conditions hold, we can generate an assignment of resources and machines to the jobs such that each machine and each resource part is allocated by at most one job at a time, see [27]. The objective is to find a feasible schedule, which minimizes the total length of the schedule called makespan, i.e., we have to minimize $\max_{j\in\mathcal{J}} \sigma(j) + p(j)$.

This problem arises naturally, e.g., in parallel computing, where jobs that are scheduled in parallel share common memory or in production logistics where different jobs need a different number of people working on it. From a theoretical perspective these problem is a sensible generalization of problems like scheduling on identical machines, parallel task scheduling and bin packing with cardinality constraint.

The algorithm with the best absolute ratio so far is a $(2+\varepsilon)$-approximation by Niemeier and Wiese [31]. In this paper, we close this gap between approximation and lower bound by presenting an algorithm with approximation ratio $(3/2+\varepsilon)$.

▶ **Theorem 1.** *There is an algorithm for single resource constraint scheduling with approximation ratio $(3/2+\varepsilon)$ and running time $\mathcal{O}(n\log(1/\varepsilon)) + \mathcal{O}(n)(m\log(R)/\varepsilon)^{\mathcal{O}_\varepsilon(1)}$, where $\mathcal{O}_\varepsilon$ dismisses all factors solely dependent on $1/\varepsilon$.*

As a by-product of this algorithm, we also present an algorithm, which has an approximation guarantee of $(1+\varepsilon)\mathrm{OPT} + p_{\max}$. Note that we can scale each instance such that $p_{\max} = 1$ and hence we can see $p_{\max}$ as a constant independent of the instance. Algorithms with an approximation guarantee of the form $(1+\varepsilon)\mathrm{OPT} + c$ fore some constant c are called asymptotic polynomial time approximation schemes (APTAS). Note that this algorithm is a $(2+\varepsilon)$-approximation as well, but improves this ratio, in the case that $p_{\max}$ is strictly smaller than OPT and for $p_{\max} < \mathrm{OPT}/2$ improves the approximation ratio of the algorithm from Theorem 1.

▶ **Theorem 2.** *There is an APTAS for single resource constraint scheduling with an additive term $p_{\max}$ and running time $\mathcal{O}(n\log(1/\varepsilon+n)) + \mathcal{O}(n)(m\log(R)/\varepsilon)^{\mathcal{O}_\varepsilon(1)}$.*

In the schedule generated by this APTAS, almost all jobs are completed before $(1 + \mathcal{O}(\varepsilon))\mathrm{OPT}$, except for a small set $\mathcal{J}'$ of jobs that all start simultaneously at $(1+\mathcal{O}(\varepsilon))\mathrm{OPT}$, after the processing of all other jobs is finished. The processing of this set $\mathcal{J}'$ causes the additive term $p_{\max}$.

Methodology and Organization of this Paper

In Section 2, we will present the main results to generate the APTAS from Theorem 2. The general structure of the algorithm can be summarized as follows. First, we simplify the instance by rounding the processing time of the jobs and partitioning them into large, medium, and small corresponding to their processing time. Afterward, we use some linear programming approaches to find a placement of these jobs inside the optimal packing. The few jobs that are placed fractional with this linear program will be placed on top of the packing contributing to the set $\mathcal{J}'$ which was mentioned before.

As usual for this kind of algorithms for packing and scheduling problems, we divide the jobs into large, medium and small jobs. While for the placement of medium and small jobs, we use techniques already known, see e.g. [21], we used a new technique to place the large jobs. For the following $(3/2+\varepsilon)$ approximation, it is important to guarantee that only $\mathcal{O}_\varepsilon(1)$, i.e. constant in $1/\varepsilon$, large jobs are not placed inside the optimal scheduling area. To find such a schedule, we divide the schedule into $\mathcal{O}_\varepsilon(1)$ time slots and guess the machine requirement and a rounded resource requirement of large jobs during this slot. Afterward, the large jobs are placed inside this profile using a linear program, which schedules only $\mathcal{O}_\varepsilon(1)$ of these jobs fractionally. We have to remove these $\mathcal{O}_\varepsilon(1)$ fractionally scheduled jobs, as well as only one extra job per time slot due to the rounded guess, and assign them to the set $\mathcal{J}'$.

Afterward, in Section 3, we present the $(3/2+\varepsilon)$-approximation and prove Theorem 1. Instead of placing the fractional jobs on top of the packing, we stretch the packing by $(1/2+\mathcal{O}(\varepsilon))\mathrm{OPT}$, and place the fractional scheduled large jobs inside a gap in this stretched schedule. This stretching allows us to define a common finishing point of (almost) all the jobs, which have a processing time larger than OPT/2 and, thus, we avoid scheduling them fractionally with the linear program. While the general idea of creating such a gap was used before, e.g., in [19], the novelty of this approach is to search for this gap at multiple points in time while considering two and not only one constraint. This obstacle of considering two instead of one constraint while searching for a gap requires a more careful analysis of the shifted schedule, as was needed in other gap constructions.

Related Work

The problem resource constraint scheduling is one of the classical problems in scheduling. It was first studied in 1975 by Garey and Graham [10]. Given m identical machines and capacities $R_1, \dots, R_s$ of s distinct resources such that each job requires a share of each of them, they proved that the greedy list algorithm produces a schedule of length at most $(s + 2 - (2s + 1)/m)\mathrm{OPT}$. This corresponds to an approximation ratio of $(3 - 3/m)$ for the case of $s = 1$ i.e., the problem studied in this paper. In the same year Garey and Johnson [11] showed that this general scheduling problem is NP-complete even if just one resource is given, i.e., $s = 1$. Lately, Niemeier and Wiese [31] presented a $(2 + \varepsilon)$-approximation for single resource constraint scheduling, and this is the best known ratio so far.

Note that the problem single resource constraint scheduling contains multiple problems as subproblems. When all the processing times are equal to one, this problem corresponds to bin packing with cardinality constraint. Hence there is no algorithm with an approximation guarantee better than $3/2$ for this problem unless $\mathrm{P} = \mathrm{NP}$. For bin packing with cardinality constraint Epstein and Levin [8] presented an AFPTAS. This AFPTAS was improved and extended to work for single resource constraint scheduling by Jansen et al. [21]. It has an additive term of $\mathcal{O}(p_{\max} \log(1/\varepsilon)/\varepsilon)$.

On the other hand, if the number of machines m is larger than n, the constraint that only m jobs can be processed at the same time is no longer a restriction. The resulting problem is known as the parallel task scheduling problem. This problem is strongly NP-complete for $R \geq 4$ [17] and there exists a pseudo polynomial algorithm for $R \leq 3$ [7]. Furthermore, for R constant and $R \in n^{\mathcal{O}(1)}$ there exists polynomial time approximation schemes by Jansen and Porkolab [22] as well as Jansen and Thöle [26] respectively. For an arbitrary large R there exists no algorithm with approximation ratio smaller than $3/2$ unless $P = NP$. The best algorithm for this scenario is a $(3/2 + \varepsilon)$ approximation by Jansen [19].

Finally, if each job requires at most R/m from the resource, the used resources are no longer a restriction. The corresponding problem is known as the NP-hard problem makespan scheduling on identical machines. For this problem several algorithms are known, see, e.g., [13, 18, 2, 20].

An interesting extension of the considered problem is the consideration of resource dependent processing times. In this scenario, instead of a fixed resource requirement for the jobs, each jobs processing time depends on the amount of allocated resources. The first result for this extension was achieved by Grigoriev et al. [14]. They studied a variant where the processing time of a job depends on the machine it is processed on as well as the number of assigned resources and described a 3.75-approximation. For the case of identical machines this result was improved by Kellerer [28] to a $(3.5 + \varepsilon)$-approximation. Finally, Jansen at al [21] presented an AFPTAS for this problem with additive term $\mathcal{O}(\pi_{\max} \log(1/\varepsilon)/\varepsilon)$, where $\pi_{\max}$ is the largest occurring processing time considering all possible resource assignments.

Closely related to single resource constraint scheduling is the strip packing problem. Here we are given a set of rectangular items that have to be placed overlapping free into a strip with bounded width and infinite height. We can interpret single resource constraint scheduling as a Strip Packing problem by setting the jobs processing time to the height of a rectangular item and the resource requirement to its width. The difference to strip packing is now, that when placing an item, we are allowed to slice it vertically as long as the lower border of all slices are placed at the same *vertical level*. Furthermore, we have an single resource constraint scheduling carnality condition that allows only m items to intersect each horizontal line through the strip. The strip packing problem has been widely studied [3, 4, 5, 6, 12, 16, 25, 29, 32, 33, 34, 35]. The algorithm with approximation ratio $(5/3 + \varepsilon)$, which is the smallest so far, was presented by Harren, Jansen, Prädel, and van Stee [15]. On the other hand, using a reduction from the partition problem, we know that there is no polynomial-time algorithm with an approximation ratio smaller than 3/2. Closing this gap between the best approximation ratio and lower bound represents an open question. Strip packing has also been studied with respect to asymptotic approximation ratio [3, 6, 12, 29]. The best algorithms in this respect are an *AFPTAS* with additive term $\mathcal{O}(1/\varepsilon \log(1/\varepsilon))h_{\max}$ [35, 5] and an APTAS with additive term $h_{\max}$ [25], where $h_{\max}$ is the tallest height in the set of given items. Finally, this problem has been studied with respect to pseudo-polynomial processing time [36, 26, 30, 9, 23, 1], where the width of the strip is allowed to occur polynomial in the running time of the algorithm. There is no pseudo-polynomial algorithm with an approximation ratio smaller than 5/4 [17] and a ratio of $(5/4 + \varepsilon)$ is achieved by the algorithm in [24].

2 APTAS with additive term $p_{\max}$

In this section, we present an asymptotic PTAS for the single resource constraint scheduling problem which has an approximation guarantee of $(1+\varepsilon)\mathrm{OPT} + p_{\max}$ and a running time of $\mathcal{O}(n\log(n)) + (m\log(R))^{\mathcal{O}_\varepsilon(1)}$, i.e., we prove Theorem 2 in this section. Due to space limitations the proofs of this section can be found in the appendix.

Simplifying the input instance. In the first step of the algorithm, we simplify the given instance such that it has a simple structure and a reduced set of processing times. Consider the lower bound on the optimal makespan $T := \min\{p_{\max}, \mathrm{area}(\mathcal{J})/R, p(\mathcal{J})/m\}$. By the analysis of the greedy list schedule, as described in [31], we know that the optimal schedule has a size of at most $\frac{1}{m}p(\mathcal{J}) + \frac{2}{R}\mathrm{area}(\mathcal{J}) + p_{\max} \leq 4T$, giving us appropriate bounds for a dynamic search framework.

In the next step, we will create a gap between jobs with a large processing time and jobs with a small processing time, by removing a set of medium sized jobs. We want to schedule this set of medium sized jobs in the beginning or end of the schedule using the greedy list schedule. However this schedule of the medium sized items should add at most $\mathcal{O}(\varepsilon)\mathrm{OPT}$ to the makespan. The schedule generated by the greedy list schedule algorithm has a makespan of at most $\frac{1}{m}p(\mathcal{J}) + \frac{2}{R}\mathrm{area}(\mathcal{J}) + p_{\max} \leq 4\mathrm{OPT}$. Hence, we choose the set of medium jobs $\mathcal{J}_M$ such that $p_{\max}(\mathcal{J}_M) \leq \varepsilon\mathrm{OPT}$, i.e. the maximal processing time appearing in the set of medium jobs is bounded by $\varepsilon\mathrm{OPT}$. On the other hand, the total area and the total processing time of the medium jobs should be small enough.

▶ **Lemma 3.** *Consider the sequence $\gamma_0 = \varepsilon$, $\gamma_{i+1} = \gamma_i\varepsilon^4$. There exists an $i \in \{1,\dots,1/\varepsilon\}$ such that*

$$\frac{1}{m}p(\mathcal{J}_{\gamma_i}) + \frac{2}{R}\mathrm{area}(\mathcal{J}_{\gamma_i}) \leq \varepsilon\left(\frac{1}{m}p(\mathcal{J}) + \frac{2}{R}\mathrm{area}(\mathcal{J})\right), \tag{1}$$

where $\mathcal{J}_{\gamma_i} := \{j \in \mathcal{J} \mid p(j) \in [\gamma_i T, \gamma_{i-1}T)\}$ and we can find this i in $\mathcal{O}(n + 1/\varepsilon)$.

Let $i \in \{1, \dots, 1/\varepsilon\}$ be the smallest value such that $\mathcal{J}_{\gamma_i}$ has the property from Lemma 3 and define $\mu := \gamma_i$ and $\delta := \gamma_{i-1}$. Note that $\gamma_i = \varepsilon^{1+4i}$ and hence $\delta \geq \varepsilon^{4/\varepsilon+1}$. Using these values for δ and μ, we partition the set of jobs into large $\mathcal{J}_L := \{j \in \mathcal{J} | p(j) \geq \delta T\}$, small $\mathcal{J}_S := \{j \in \mathcal{J} | p(j) < \mu T\}$ and medium $\mathcal{J}_M := \{j \in \mathcal{J} | \mu T \leq p(j) < \delta T\}$.

▶ **Lemma 4.** *The medium jobs can be scheduled in $\mathcal{O}(n \log(n))$ operations with makespan $\mathcal{O}(\varepsilon)T$*

The final simplification step is to round the processing times of the large jobs using the rounding in Lemma 5 to multiples of $\varepsilon\delta T$.

▶ **Lemma 5** (See [23]). *Let be $\delta \geq \varepsilon^k$ for some value $k \in \mathbb{N}$. At loss of a factor $(1 + 2\varepsilon)$ in the approximation ratio, we can round the processing time of each job j with processing time $\varepsilon^{l-1}T \geq p(j) \geq \varepsilon^l T$ for some $l \in \mathbb{N} \leq k$ such that it has a processing time $k_j\varepsilon^{l+1}T$ for a value $k_j \in \{1/\varepsilon, \dots 1/\varepsilon^2 - 1\}$. Furthermore, the jobs can be started at a multiple of $\varepsilon^{l+1}T$.*

In this step, we reduce the number of different processing times of large jobs to $\mathcal{O}(\log_\varepsilon(\delta)/\varepsilon^2) = \mathcal{O}(1/\varepsilon^3)$. However, we lengthen the schedule at most by the factor $(1 + 2\varepsilon)$. Furthermore, this rounding reduces the starting times of these jobs to at most $\mathcal{O}(1/(\varepsilon\delta))$ possibilities since all the large jobs start and end at multiples of $\varepsilon\delta T$ and the optimal makespan of the rounded instance is bounded by $(1 + 2\varepsilon) \cdot 4T$.

Scheduling Large Jobs. In this section, we describe how to schedule the large jobs when given the size of the makespan $T' := l\varepsilon T$ of the rounded schedule. For a given set $\mathcal{S}$ of start and endpoints of long jobs, we define a layer l_i as the processing time between two consecutive starting times $s_i, s_{i+1} \in \mathcal{S}$. Notice that, during the processing of a layer in a rounded optimal schedule, the resource requirement and number of machines used by large jobs stays unchanged since the large jobs only start and end at the starting points in $\mathcal{S}$.

▶ **Lemma 6.** *Let $\gamma \in (0, 1]$ and $T' = l\varepsilon T \geq \mathrm{OPT}$ for some $l \in \mathbb{N}$, and $\bar{\mathcal{J}}$ be a set of jobs for which an optimal schedule exists such that all jobs in $\bar{\mathcal{J}}$ have their starting and endpoints in $\mathcal{S}$. There exists an algorithm that finds in $\mathcal{O}((m \log(R))^{\mathcal{O}_\varepsilon(1)|\mathcal{S}|/\gamma})$ operations $\mathcal{O}((m \log(R))^{\mathcal{O}_\varepsilon(1)/\gamma})$ schedules with the following properties*

1. *In each of the schedules, all large jobs are scheduled except for a set $\mathcal{J}' \subseteq \bar{\mathcal{J}}$ of at most $|\mathcal{J}'| \in 3|\mathcal{S}|$ jobs and a total resource requirement of at most $R(\mathcal{J}') \leq \gamma R$.*
2. *In at least one of the schedules, in each layer given by $\mathcal{S}$, the total number of machines and resources not used by jobs in $\bar{\mathcal{J}}$ is as large as in a rounded optimal schedule.*

In the APTAS we will use the algorithm from Lemma 6 with $\bar{\mathcal{J}} := \mathcal{J}_L$ and $\mathcal{S}' := \mathcal{S}$.

Scheduling Small Jobs. We will schedule small jobs inside the layers using the residual free resources and machines given by the guess for the large jobs. We define $m_s^{(S)}$ as the number of machines in layer s not used by jobs with processing times larger than δT and analogously define $R_s^{(S)}$ as the number of resources not used by jobs with processing times larger than δT during the processing of this layer in an optimal schedule.

▶ **Lemma 7.** *Define for each layer $s \in \mathcal{S}$ a box with processing time $(1 + \varepsilon)\varepsilon\delta T$, $m_s^{(S)}$ machines and $R_s^{(S)}$ resources, where the values $m_s^{(S)}$ and $R_s^{(S)}$ are at least as large as in a rounded optimal schedule. There exists an algorithm with time complexity $\mathcal{O}(n) \cdot \mathcal{O}_{\varepsilon,|\mathcal{S}|}(1)$, that places the jobs inside the boxes and an additional horizontal box with m machines, R resources, and processing time $\mathcal{O}(\varepsilon)T$.*

This algorithm uses the same techniques as the AFPTAS designed by Jansen et al. [21]. For the sake of completeness the ideas can be found in the appendix.

The AFPTAS. We can summarize the algorithm as follows. We define $T := \min\{p_{\max}, \text{area}(\mathcal{J})/R, p(\mathcal{J})/m\}$ and simplify the instance as described above. Then via a binary search framework, we try values $T' \in [T, 4T]$ as optimal makespan. For each of the considered values T', we use the algorithm from Lemma 6 to generate several scheduled for the large jobs each wit makespan at most T'. If we cannot find a feasible schedule, the value T' was to small. Otherwise, we use the algorithm from Lemma 7 to schedule the small jobs inside each of the generated schedules for the large jobs. If T' was large enough, we can place the small jobs inside the layers, by increasing the schedule by a factor of at most $\mathcal{O}(\varepsilon)$. If the small jobs do not fit in one of the schedules for large jobs, the chosen T' was to small. If we have found a schedule for the small jobs, we try the next smaller value for T' in binary search fashion. In the final step, we use greedy list schedule to schedule the medium jobs and place the set $\mathcal{J}'$ of non scheduled large jobs at the top.

3 A $(3/2 + \varepsilon)$-Approximation

We aim to find a schedule with makespan $(3/2 + \mathcal{O}(\varepsilon))T'$, where T' is the assumed optimal makespan given by a binary search framework. Consider Lemma 6. If one of the jobs in $\mathcal{J}'$ has a processing time larger than $T'/2 + \mathcal{O}(\varepsilon)T$, we exceed the aspired approximation ratio of $(3/2 + \mathcal{O}(\varepsilon))T$, when placing the set $\mathcal{J}'$ on top of the schedule. We call this set of critical jobs huge jobs, i.e., $\mathcal{J}_H := \{j \in \mathcal{J} | p(j) > T'/2\}$, and redefine the set of large jobs as $\mathcal{J}_L := \{j \in \mathcal{J} | \delta T \leq p(j) \leq T'/2\}$ respectively.

Notice that the processing of all huge jobs has to intersect the time $T'/2$ in each schedule with makespan at most T' and each machine can contain at most one of these jobs. If we could guess the starting positions of these huge jobs, and schedule only the large jobs with the algorithm from Lemma 6, the discarded jobs $\mathcal{J}'$ would have a processing time of at most $T'/2$ and could be placed on top of the schedule, resulting in a schedule of makespan at most $(3/2 + \mathcal{O}(\varepsilon))T'$. Sadly this guessing step is not possible in polynomial time since there are up to m of these jobs and iterating all combinations of their starting position needs $\Omega((1/\varepsilon\delta)^m)$ operations. Our idea is to let almost all the huge jobs end at a common point in time, e.g. T', and thus avoid the guessing step. To solve the violation of the resource or machine condition, we shift up all the jobs which start after $\lceil T'/2 \rceil_{\varepsilon\delta T}$ by $\lceil T'/2 \rceil_{\varepsilon\delta T}$ such that they now start after T', where we denote by $\lceil T'/2 \rceil_{\varepsilon\delta T}$ the integer multiple of $\varepsilon\delta T$ that is the first which has a size of at least $T'/2$.

While this shift fixes the start positions of the huge jobs, the large jobs are again placed with the techniques described in Section 2. Since Lemma 6 states that each of the generated schedules may not schedule a subset $\mathcal{J}'$ of the large jobs, we need to find a gap in the shifted schedule where we can place them. In the following, we will consider optimal schedules and the possibilities to rearrange the jobs. Depending on this arrangement, we can find a gap of processing time $\lceil T'/2 \rceil_{\varepsilon\delta T}$ for the fractional scheduled large jobs.

Let us assume that we have to schedule $k := |\mathcal{J}'| \in \mathcal{O}_\varepsilon(1) \leq m/4$ jobs with total resource requirement at most $\gamma R \leq R/(3|\mathcal{S}|)$. We consider an optimal schedule, after applying the simplification steps and the corresponding transformed optimal schedule, where each large job starts at a multiple of $\varepsilon\delta T$ and each huge job starts at a multiple of $\varepsilon^2 T$. Furthermore, we will assume that there are more than $4k = \mathcal{O}_\varepsilon(1)$ huge jobs. Otherwise, we can guess their starting positions in $\mathcal{O}((1/\varepsilon\delta)^{4k})$ and place the fractional scheduled large jobs on top of the schedule.

In the following, we will prove that by extending it by $\lceil T'/2 \rceil_{\varepsilon\delta T}$, we can transform the rounded optimal schedule $\text{OPT}_{\text{rounded}}$ such that all the huge jobs, except for $\mathcal{O}(k)$ of them, end at a common point in time and we can place k further narrow large jobs without violating the machine or the resource constraint.

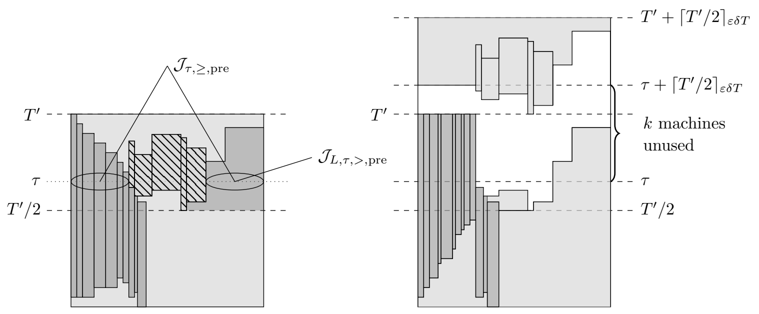

$\mathcal{J}_{\tau,\geq,\text{pre}}$
T'
τ
$T'/2$
$\mathcal{J}_{L,\tau,>,\text{pre}}$
T'
$T' + \lceil T'/2\rceil_{\varepsilon\delta T}$
$\tau + \lceil T'/2\rceil_{\varepsilon\delta T}$
k machines unused
τ
$T'/2$

Figure 1 In this and the following figures we present the processing time on the y-axis, while the resource requirements of jobs can be found on the x-axis. While the machines are not visually represented in these figures, the machine condition has to apply for each horizontal cut through the schedule. **On the left:** A rounded optimal schedule. The hatched rectangles are the jobs that start after $T'/2$ and intersect τ, the dark gray area corresponds to large jobs, which start before $T'/2$ and end after τ and the dark gray rectangles on the left are huge jobs. **On the right:** The corresponding shifted schedule.

▶ **Lemma 8.** *Let $k := |\mathcal{J}'| \leq m/4$ and $\gamma \leq 1/(3|\mathcal{S}|)$. Furthermore, let a rounded optimal schedule $\text{OPT}_{\text{rounded}}$ with makespan at most T' and at most $|\mathcal{S}|$ starting positions for large jobs be given.*

Without removing any job from the schedule, we can find a transformed schedule $\text{OPT}_{\text{shift}}$ with makespan at most $T' + \lceil T'/2\rceil_{\varepsilon\delta T}$, with the following properties:

1. *We can guess the end positions of all huge jobs in polynomial time.*
2. *There is a gap of processing time $\lceil T'/2\rceil_{\varepsilon\delta T}$ with k empty machines and γR free resources where we can schedule the jobs in $\mathcal{J}'$.*
3. *There is an injection which maps each layer s in $\text{OPT}_{\text{rounded}}$ with $m_{s,S}$ machines and $R_{s,S}$ resources not used by huge and large jobs to a layer in $\text{OPT}_{\text{shift}}$ where there are at least as many machines and resources not used by these jobs.*

Proof. We will prove this lemma by a careful analysis of the structure of the schedule $\text{OPT}_{\text{rounded}}$. First, however, we introduce some notations. Let $s \in \mathcal{S}$, with $s > T'/2$ be any starting point of large jobs. We say a job $j \in \mathcal{J}$ intersects s or is intersected by s if $\sigma(j) < s < \sigma(j) + p(j)$. We will differentiate sets of jobs that start before $T'/2$ and those that start at or after $T'/2$ by adding the attribute $_{\text{pre}}$ to sets of jobs that contain only jobs starting before $T'/2$, and the attribute $_{\text{post}}$ to those that contain only jobs that start at or after $T'/2$. Furthermore, we will identify the sets of jobs that intersect certain points of time. We will add the attribute $_{s,\geq}$ to denote a set of jobs that is processed at least until the point in time $s \in \mathcal{S}$, i.e., we denote by $\mathcal{J}_{s,\geq,\text{pre}} := \{j \in \mathcal{J} | p(j) \geq \delta T, \sigma(j) < T'/2, \sigma(j) + p(j) \geq s\}$ the set of large and huge jobs starting before $T'/2$ and ending at or after s. On the other hand, if we are only interested in the jobs that intersect the time s, we add the attribute $_{s,>}$ to the set and mean $\mathcal{J}_{s,>,\text{pre}} := \{j \in \mathcal{J} | p(j) \geq \delta T, \sigma(j) < T'/2, \sigma(j) + p(j) > s\}$. Finally, we will indicate if the set contains only huge or only large jobs, by adding the attribute $_H$ or $_L$.

Let $\tau \in \{s | s \in S, T'/2 \leq s \leq T'\}$ be the smallest value such that there are at most $m - k$ jobs (huge or large) that start before $T'/2$ and intersect τ, i.e., end after τ. We partition the set of large jobs intersected by τ into two sets. Let $\mathcal{J}_{L,\tau,>,\text{pre}} := \{j \in \mathcal{J}_L | \sigma(j) < T'/2, \sigma(j) + p(j) > \tau\}$ be the set of large jobs which start before $T'/2$ and end at or after τ. Further let $\mathcal{J}_{L,\tau,>,\text{post}} := \{j \in \mathcal{J}_L | T'/2 \leq \sigma(j) < \tau, \sigma(j) + p(j) > \tau\}$ be the set of large jobs, which are started at or after $T/2$ but before τ and end after τ, see Figure 1.

Note that by the choice of τ in each point between τ and $T'/2$ in the schedule there are more than $m-k$ machines used by $\mathcal{J}_{\tau,\geq,\mathrm{pre}}$. As a result there are at most $k-1$ machines used by jobs starting after $T'/2$ at each point between $T'/2$ and τ, implying $|\mathcal{J}_{L,\tau,>,\mathrm{post}}| < k$.

We now construct a shifted schedule. Starting times in this schedule will be denoted by σ'. We shift each job $j \in \mathcal{J}$ with $\sigma(j) \geq T'/2$ and $\sigma(j) + p_j \geq \tau$ exactly $\lceil T'/2 \rceil_{\varepsilon\delta T}$ upwards, i.e., we define $\sigma'(j) := \sigma(j) + \lceil T'/2 \rceil_{\varepsilon\delta T}$ for these jobs. Furthermore, each huge job $j \in \mathcal{J}_H$ intersecting τ is shifted upwards such that it ends at T', i.e., we define $\sigma'(j) := T' - p_j$ for these jobs j, see Figure 1. Note that there are at most k huge jobs ending strictly before τ. If the total number of huge jobs ending before or at τ is larger than k, we choose arbitrarily from the set of jobs ending at τ and shift them until there are exactly k huge jobs ending before or at τ.

▷ Claim 9. After this shift there are at least k machines at each point between τ and $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$ that are not used by any other job.

Proof. Up to T', there are k free machines, because there is no new job starting between τ and T' since we shifted all of them up such that they start after $\lceil T'/2 \rceil_{\varepsilon\delta T}$. On the other hand, only jobs from the set $\mathcal{J}_{L,\tau,>,\mathrm{post}}$ are processed between T' and $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$. Since $|\mathcal{J}_{L,\tau,>,\mathrm{post}}| < k$ and $m - k \geq k$ this leaves k free machines which proves the claim. ◁

The idea is to place the gap at τ since there are enough free machines. However, it can happen that at a point between τ and $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$ there is not enough free resource for the gap. In the following, we carefully analyze where we can place the k jobs, dependent on the structure of the optimal schedule $\mathrm{OPT}_{\mathrm{rounded}}$

Case 1: $r(\mathcal{J}_{\tau,\geq,\mathrm{pre}}) \leq R - \gamma R$. In this case there are at least γR free resources at each point in the shifted schedule between τ and T' since there are no jobs starting between these points of time. To place the k fractional scheduled jobs, we have to generate a gap of processing time $\lceil T'/2 \rceil_{\varepsilon\delta T}$. In this gap there have to be k unused machines and γR unused resources. For the time between τ and T', we have this guarantee, while for the time between T' and $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$, we have k free machines, but might have less than γR free resource. The only jobs overlapping in this time window are the jobs from the set $\mathcal{J}_{L,\tau,>,\mathrm{post}}$, see Figure 1. If these jobs have a small resource requirement, we have found our gap, see Case 1.1. and, otherwise, we have to look more careful at the schedule.

Case 1.1: $r(\mathcal{J}_{L,\tau,>,\mathrm{post}}) \leq R - \gamma R$. In this case, the required gap is positioned between τ and $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$, see Figure 2. In this shifted optimal schedule there are at most k huge jobs ending before τ. In the algorithm, we will guess τ dependent on a given solution for the large jobs and guess these k huge jobs and their start points in $\mathcal{O}(m^k S^k)$, which is polynomial in the input size, see Section 3 for an overview.

Case 1.2: $r(\mathcal{J}_{L,\tau,>,\mathrm{post}}) > R - \gamma R$. In this case, there is a point $t \in [\tau, T']$ such that after this point there are less than γR free resources. Therefore, we need another position to place the fractionally scheduled jobs. We partition the set $\mathcal{J}_{L,\tau,>,\mathrm{post}}$ into at most $|\mathcal{S}|/2$ sets $\mathcal{J}^{\iota}_{L,\tau,>,\mathrm{post}}$ by the their original finishing points $\iota \in \mathcal{S}_{>\tau}$, i.e., each job in $\mathcal{J}^{\iota}_{L,\tau,>,\mathrm{post}}$ finishes at ι in the non shifted rounded optimal schedule.

▷ Claim 10. One of the sets $\mathcal{J}^{\iota}_{L,\tau,>,\mathrm{post}}$, $\iota \in \mathcal{S}_{>\tau}$, has a resource requirement of at least γR.

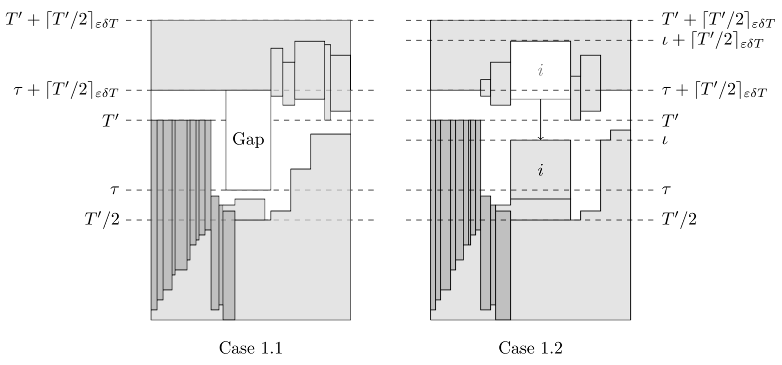

$T' + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$
T'
Gap
τ
$T'/2$
Case 1.1
$T' + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\iota + \lceil T'/2 \rceil_{\varepsilon\delta T}$
i
$\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$
T'
ι
i
τ
$T'/2$
Case 1.2

Figure 2 Examples for the two Cases 1.1. and 1.2. In Case 1.1 the gap is positioned between τ and $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$. In Case 1.2 the jobs in the set $\mathcal{J}^{\iota}_{L,\tau,>,\text{post}}$ are shifted back down. At each point between ι and $\iota + \lceil T'/2 \rceil_{\varepsilon\delta T}$ there are at least γR unused resources.

Proof. The jobs in $\mathcal{J}_{L,\tau,>,\text{post}}$ use more than $R - \gamma R$ resource in total. Since $\gamma \leq 1/(3|\mathcal{S}|) \leq 1/(|\mathcal{S}|/2 - 1)$, it holds that

$$\frac{R - \gamma R}{|\mathcal{S}|/2} \geq \frac{(1 - 1/(|\mathcal{S}|/2 - 1))R}{|\mathcal{S}|/2} = R/(|\mathcal{S}|/2 - 1) \geq \gamma R.$$

Hence, by the pigeon principle, one of the sets, say $\mathcal{J}^{\iota}_{L,\tau,>,\text{post}}$, must have a resource requirement of at least γR. ◁

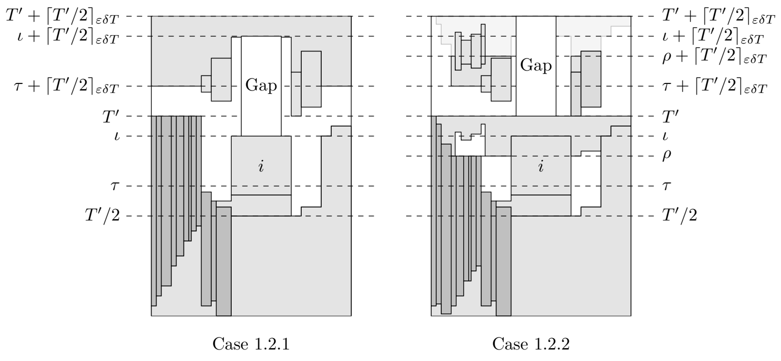
Let $\mathcal{J}^{\iota}_{L,\tau,>,\text{post}}$ be this set. To generate a gap, we shift down all jobs in $\mathcal{J}^{\iota}_{L,\tau,>,\text{post}}$ back to their primary start position, see Figure 2.

▷ Claim 11. As a result of this shift, there are at least γR free resources at each point between ι and $\iota + \lceil T'/2 \rceil_{\varepsilon\delta T}$.

Proof. At each point between ι and T' there were γR unused resources before. Each job which starts between T' and $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$ is an element of $\mathcal{J}_{L,\tau,>,\text{post}}$ and was therefore scheduled in parallel to the jobs in $\mathcal{J}^{\iota}_{L,\tau,>,\text{post}}$. Therefore, at each point between T' and $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$ at least γR resources are unused. From $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$ to $\iota + \lceil T'/2 \rceil_{\varepsilon\delta T}$ the jobs $\mathcal{J}^{\iota}_{L,\tau,>,\text{post}}$ were scheduled, so there are at least γR free resources. ◁

We now have to differentiate if there are at least k machines unused between $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$ and $\iota + \lceil T'/2 \rceil_{\varepsilon\delta T}$, see Figure 2. Let $\rho \in \{s | \tau \leq s \leq T, s \in S\}$ be the first point in the schedule where at most k jobs from $\mathcal{J}_{\tau,\geq,\text{pre}}$ are scheduled in the given optimal schedule (not the shifted one), i.e., ρ is the first point in time where $|\mathcal{J}_{\rho,>,\text{pre}}| \leq k$. Note that as a consequence $|\mathcal{J}_{\rho,\geq,\text{pre}}| \geq k$ since otherwise there would have been a point in time before ρ, where at most k machines are used by jobs starting before $T'/2$. We know that between T' and $\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$ there always will be k machines unused since before the first shift they were blocked by jobs in $\mathcal{J}_{\tau,\geq,\text{pre}}$.

Case 1.2.1: $\rho \geq \iota$. In this case, at each point between ι and $\iota + \lceil T'/2 \rceil_{\varepsilon\delta T}$ there are k machines unused. Between ι and T' there are k free machines by the choice of τ and between T' and $\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$ there are k free machines by the choice of ρ. Therefore, there is a gap between ι and $\iota + \lceil T'/2 \rceil_{\varepsilon\delta T}$, see Figure 3. Similar as in Case 1.1. the total number of guesses needed to place the huge jobs is bounded by $m_{\varepsilon}^{\mathcal{O}}(1)$, although we have to add the guess for ρ.



$T' + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\iota + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$
T'
ι
τ
$T'/2$
Gap
i
Case 1.2.1
$T' + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\iota + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$
T'
ι
ρ
τ
$T'/2$
Gap
i
Case 1.2.2

Figure 3 Examples for the shifted schedule and the position of the gap in the Cases 1.2.1 and 1.2.2.

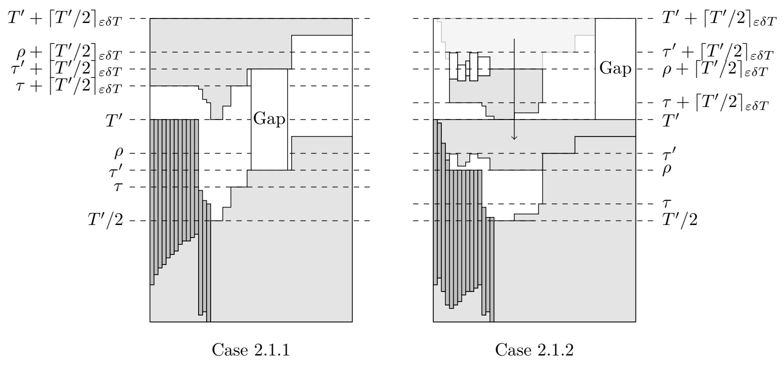
Case 1.2.2: $\rho < \iota$. Let $\mathcal{J}_{H,\rho} := \{j \in \mathcal{J}_H | s_j + p_j > \rho\}$ be the set of huge jobs, which are still scheduled after ρ. It holds that $|\mathcal{J}_{H,\rho}| \leq k$. As a consequence, it is possible to guess their starting positions in polynomial time. Therefore, the algorithm will schedule each job in $\mathcal{J}_{H,\rho}$ as in the original simplified schedule $\mathrm{OPT}_{\mathrm{rounded}}$. The other huge jobs, which end between τ and ρ, are scheduled such that they end at ρ, i.e., we define $\sigma'(j) := \rho - p(j)$ for each of these huge jobs j. Next, we shift the all the jobs j with starting time $\sigma'(j) \geq \rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$ downwards such that they start as they had started before the first shift. As a result between T' and $T' + \lceil T'/2 \rceil_{\varepsilon\delta T}$, there are just jobs left which overlap the time from $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$ to $\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$, see Figure 3. By the choice of ρ and τ at each point between $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$ and $\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$ there are at most $m - k$ jobs which use at most $R - \gamma R$ resource since the job i was scheduled there before. Since each job between T' and $T' + \lceil T'/2 \rceil_{\varepsilon\delta T}$ overlaps this area there are at least k free machines and γR free resources in this area. Hence, we position the gap at T'. In the algorithm, we will guess τ and ρ dependent on a given fractional solution for the large jobs and guess the at most k jobs ending before τ and the k jobs ending after ρ in $\mathcal{O}(m^{2k-1})$. For each of these jobs, we have to guess its starting time out of at most $|\mathcal{S}|/2$ possibilities.

Case 2: $r(\mathcal{J}_{\tau,\geq,\mathrm{pre}}) > R - \gamma R$. In this case, the gap has to start strictly after τ since at τ there is not enough free resource. Let $\mathcal{J}_{L,T'/2}$ be the set of large jobs intersecting the point in time $T'/2$. Remember that $\mathcal{J}_{\tau,\geq,\mathrm{pre}}$ contains huge and large jobs. Since $r(\mathcal{J}_{\tau,\geq,\mathrm{pre}}) > R - \gamma R$ at least one of these stets of jobs (huge or large) has to contribute a large resource requirement to $r(\mathcal{J}_{\tau,\geq,\mathrm{pre}})$. In the following, we will find the gap, depending on which of both sets contributes a suitable large resource requirement.

Case 2.1: $r(\mathcal{J}_{L,T'/2}) \geq 2\gamma R$. Let $\tau' \in \{s \in S | \tau \leq s \leq T'\}$ be the first point in time where $r(\mathcal{J}_{L,T'/2}) - r(\mathcal{J}_{L,\tau',>,\mathrm{pre}}) \geq \gamma R$. Note that $\tau \leq \tau'$ since, otherwise, there would be γR free resources at τ.

▷ Claim 12. By this choice at each point between τ' and $\tau' + \lceil T'/2 \rceil_{\varepsilon\delta T}$ there are at least γR free resources.

Between τ' and T' there are γR free resources since jobs form $\mathcal{J}_{L,T'/2}$ with a resource requirement of at least γR end before τ'. On the other hand, before the shift there was at least γR resource blocked by jobs from $\mathcal{J}_{L,T'/2}$ between $T'/2$ and τ' and hence after the shift there is at least γR free resource at any time between T' and $\tau' + \lceil T'/2 \rceil_{\varepsilon\delta T}$. Moreover, as in Case1.2, let $\rho \in \{s | \tau \leq s \leq T', s \in S\}$ be the first point in the schedule where $|\mathcal{J}_{\rho,>,\mathrm{pre}}| \leq k$, i.e., where at most k jobs are scheduled that start before $T'/2$.



$T' + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\tau' + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$
T'
ρ
τ'
τ
$T'/2$
Gap
Case 2.1.1
Gap
$T' + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\tau' + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$
T'
τ'
ρ
τ
$T'/2$
Case 2.1.2

Figure 4 Examples for the shifted schedules and the position of the gap in Cases 2.1.1 and Case 2.1.2.

▷ Claim 13. By this choice at each point between τ and $\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$ there are at least k unused machines.

From τ to T' there are k unused machines, by the choice of τ. On the other hand, at each point in time between T' and $\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$ there where k machines blocked by jobs from that started before $T'/2$ and these machines are now unused.

Similar as in Cases 1.2.1 and 1.2.2, we will find the gap dependent of the relation between τ' and ρ.

Case 2.1.1: $\rho \geq \tau'$. In this case between τ' and $\tau' + \lceil T'/2 \rceil_{\varepsilon\delta T}$ there are at least k unused machines. Therefore, we have a gap between these two points, which is large enough, see Figure 4. In the algorithm, we have to guess the k huge jobs, which end before τ and their start point, as well as the points τ, τ' and ρ. All the guesses for this case can be iterated in polynomial time.

Case 2.1.1: $\rho < \tau'$. In this case, we act like in Case 1.2.2 and shift all huge jobs, but the at most k jobs ending after ρ, downwards such that they end at ρ, see Figure 4. Furthermore, we shift all jobs starting after $\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$ back downwards such that they again start at their primary start position. Now after T' there are just jobs having their start or end position between $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$ and $\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$. At each point between these two points there are at least k unused machines and γR unused resource with the same arguments as in Case 1.2.2. Hence, we have a gap with the right properties between T' and $T' + \lceil T'/2 \rceil_{\varepsilon\delta T}$. All the guesses for this case can be iterated in polynomial time.

Case 2.2: $r(\mathcal{J}_{L,T'/2}) < 2\gamma R$. Since we have $r(\mathcal{J}_{\tau,\geq,\text{pre}}) > R - \gamma R$ (by Case 2.) and it holds that $(\mathcal{J}_H \cap \mathcal{J}_{\tau,\geq,\text{pre}}) \cup (\mathcal{J}_{L,T'/2} \cap \mathcal{J}_{\tau,\geq,\text{pre}}) = \mathcal{J}_{\tau,\geq,\text{pre}}$ we get that $r(\mathcal{J}_H \cap \mathcal{J}_{\tau,\geq,\text{pre}}) \geq R - 3\gamma R$. Similar as before, let $\rho \in \{s | \tau \leq s \leq T, s \in S\}$ be the first point in the schedule where less than k jobs are scheduled that start before $T'/2$. By the same argument as in Case 2.1, we know that at every point between τ and $\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$ there are at least k unused machines in the shifted schedule.

Case 2.2.1: $r(\mathcal{J}_{\rho,\geq,\text{pre}}) \geq \gamma R$. In this case, we can construct a schedule in the same way as in case 1.2.2 or 2.1.2 by shifting down the jobs that start after $\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$ and positioning the gap at T', see Figure 5. This is possible because the jobs that are scheduled

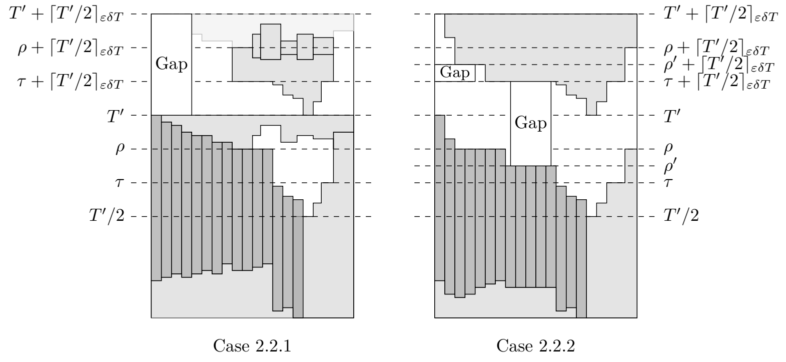

$T' + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$
T'
ρ
τ
$T'/2$
Gap
Case 2.2.1
$T' + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\rho' + \lceil T'/2 \rceil_{\varepsilon\delta T}$
$\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$
T'
ρ
ρ'
τ
$T'/2$
Gap
Gap
Case 2.2.2

Figure 5 The shifted schedule and the position of the gap in Cases 2.2.1 and 2.2.2. Note that in Case 2.2.2 the gap is not displayed continuously. However, by swapping the used resource, we can make it continuous. We only need the fact that at each point in time there are enough free resources and machines.

between $\tau + \lceil T'/2 \rceil_{\varepsilon\delta T}$ and $\rho + \lceil T'/2 \rceil_{\varepsilon\delta T}$ can use at most $R - \gamma R$ resources in this case since $r(\mathcal{J}_{\rho,\geq,\mathrm{pre}}) \geq \gamma R$ and hence at least γR resources are blocked by the jobs in $\mathcal{J}_{\rho,\geq,\mathrm{pre}}$. All the guesses for this case can be iterated in polynomial time.

Case 2.2.2: $r(\mathcal{J}_{\rho,\geq,\mathrm{pre}}) < \gamma R$. Let $\rho' \in \{i\delta^2 | \tau/\delta^2 \leq i \leq \rho/\delta^2, i \in \mathbb{N}\}$ be the smallest value, where $r(\mathcal{J}_{\rho',\geq,\mathrm{pre}}) \leq \gamma R$. Remember, we had $r(\mathcal{J}_H \cap \mathcal{J}_{\tau,\geq,\mathrm{pre}}) \geq R - 3\gamma R$ so huge jobs with summed resource requirement of at least $R - 4\gamma R$ are finished till ρ'. We partition the huge jobs that finish between τ and ρ by their processing time. Since each job has a processing time of at least $\lceil T'/2 \rceil_{\varepsilon\delta T}$, we get at most $\mathcal{O}(1/\varepsilon\delta) \leq |\mathcal{S}|/2$ sets. As seen in Section 2, we have to discard at most $k \leq 3|\mathcal{S}|$ large jobs, which have to be placed later on.

▷ Claim 14. There exists a set in the partition, which uses at least $3\gamma R$ resource total.

Proof. Since $\gamma \leq 1/(2|\mathcal{S}|) \leq 1/(3|\mathcal{S}|/2 + 4)$ it holds that

$$\frac{R - 4\gamma R}{|\mathcal{S}|/2} \geq \frac{(1 - 4/(3|\mathcal{S}|/2 + 4))R}{|\mathcal{S}|/2} = 3R/(3|\mathcal{S}|/2 + 4) \geq 3\gamma R.$$

Therefore, by the pigeon principle, there must be one set in the partition, which has summed resource requirement of at least $3\gamma R$. ◁

We sort the jobs in this partition by non increasing order of resource requirement. We greedily take jobs from this set, till they have a summed resource requirement of at least γR and schedule them such that they end before ρ'. If there was a job with more than γR resource requirement, it had to be finished before ρ' since the resource requirement of huge jobs finishing after ρ' is smaller than γR and we only chose it. Otherwise, the greedily chosen jobs have summed resource requirement of at $2\gamma R$. Since the considered set has a summed resource requirement of at least $3\gamma R$, jobs of this set with summed resource requirement at least $2\gamma R$ end before ρ'. Therefore, we do not violate any constraint by shifting down these jobs such that they end at ρ', see Figure 5.

Concerning property three, note that since we only use the free area (machines and resources) to schedule the k large jobs inside the gap, there is a layer s' in the shifted schedule, for each layer $s \in \mathcal{S}$ that has at least as many machines and resources not used by large and huge jobs as the layer s. ◀

Algorithm Summary. Given a value $T' := i\varepsilon' T$, we determine the set $\mathcal{S}$ and call the algorithm from Lemma 6 with $\gamma = 1/(3|\mathcal{S}| + 4)$ to generate the set of schedules for the large jobs. One of these schedules uses in each layer at most as many machines and resources for large jobs, as the rounded optimal schedule, or the value T' is to small. Furthermore, the set of not scheduled large jobs $\mathcal{J}'$ has a total machine requirement of at most $3|\mathcal{S}|$, a total resource requirement of at most γR, and each job has a processing time of at most $T'/2$.

For each of these schedules, the algorithm iterates all values for τ and ρ and all possibilities for the at most $2k$ huge jobs ending before or after these values and their starting positions. Then, we identify the case and the other variables dependent on the guesses and the solution schedule for the large jobs. By this we generate a new set of schedules, for which it will try to place the small jobs. We refer to the full version for more details.

To bound the total number of guesses that we add by this procedure, note that we have to guess τ, ρ and ρ' from at most $\mathcal{O}(|\mathcal{S}|)$ possibilities. Further, we for each of these guesses, the algorithm guesses at most $2k$ huge jobs and their starting positions. The total number of these guesses is bounded by $(m/\varepsilon^2)^{\mathcal{O}(k)}$, since the huge jobs start at multiples of $\varepsilon^2 T$. Therefore, the total number of guesses for the large jobs is bounded by $(m/\varepsilon^2)^{\mathcal{O}(k)} \cdot \mathcal{O}(|\mathcal{S}|^3)$. Since $k \leq 3|\mathcal{S}|$, this guess for the huge jobs lengthens the running time by a factor of at most $(m/\varepsilon)^{1/\varepsilon^{\mathcal{O}(1/\varepsilon^2)}}$. This concludes the proof of Theorem 1.

References

1 Anna Adamaszek, Tomasz Kociumaka, Marcin Pilipczuk, and Michal Pilipczuk. Hardness of approximation for strip packing. *TOCT*, 9(3):14:1–14:7, 2017. doi:10.1145/3092026.

2 Noga Alon, Yossi Azar, Gerhard J Woeginger, and Tal Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.

3 Brenda S. Baker, Donna J. Brown, and Howard P. Katseff. A 5/4 algorithm for two-dimensional packing. *Journal of Algorithms*, 2(4):348–368, 1981. doi:10.1016/0196-6774(81)90034-1.

4 Brenda S. Baker, Edward G. Coffman Jr., and Ronald L. Rivest. Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4):846–855, 1980. doi:10.1137/0209064.

5 Marin Bougeret, Pierre-François Dutot, Klaus Jansen, Christina Robenek, and Denis Trystram. Approximation algorithms for multiple strip packing and scheduling parallel jobs in platforms. *Discrete Mathematics, Algorithms and Applications*, 3(4):553–586, 2011. doi:10.1142/S1793830911001413.

6 Edward G. Coffman Jr., Michael R. Garey, David S. Johnson, and Robert Endre Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980. doi:10.1137/0209062.

7 Jianzhong Du and Joseph Y.-T. Leung. Complexity of scheduling parallel task systems. *SIAM Journal on Discrete Mathematics*, 2(4):473–487, 1989. doi:10.1137/0402042.

8 Leah Epstein and Asaf Levin. AFPTAS results for common variants of bin packing: A new method for handling the small items. *SIAM Journal on Optimization*, 20(6):3121–3145, 2010. doi:10.1137/090767613.

9 Waldo Gálvez, Fabrizio Grandoni, Salvatore Ingala, and Arindam Khan. Improved pseudo-polynomial-time approximation for strip packing. In *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 9:1–9:14, 2016. doi:10.4230/LIPIcs.FSTTCS.2016.9.

10 Michael R. Garey and Ronald L. Graham. Bounds for multiprocessor scheduling with resource constraints. *SIAM Journal on Computing*, 4(2):187–200, 1975. doi:10.1137/0204015.

11 Michael R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397–411, 1975.

12 Igal Golan. Performance bounds for orthogonal oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 10(3):571–582, 1981. doi:10.1137/0210042.

13 Ronald L Graham. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics 17.2*, pages 416–429, 1969.

14 Alexander Grigoriev, Maxim Sviridenko, and Marc Uetz. Machine scheduling with resource dependent processing times. *Mathematical programming*, 110(1):209–228, 2007.

15 Rolf Harren, Klaus Jansen, Lars Prädel, and Rob van Stee. A $(5/3 + \epsilon)$-approximation for strip packing. *Computational Geometry*, 47(2):248–267, 2014. doi:10.1016/j.comgeo.2013.08.008.

16 Rolf Harren and Rob van Stee. Improved absolute approximation ratios for two-dimensional packing problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques,*, volume 5687 of *Lecture Notes in Computer Science*, pages 177–189. Springer, 2009. doi:10.1007/978-3-642-03685-9_14.

17 Sören Henning, Klaus Jansen, Malin Rau, and Lars Schmarje. Complexity and inapproximability results for parallel task scheduling and strip packing. *Theory of Computing Systems*, 2019. doi:10.1007/s00224-019-09910-6.

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

19 Klaus Jansen. 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)*, pages 224–235, 2012. doi:10.1145/2312005.2312048.

20 Klaus Jansen, Kim-Manuel Klein, and José Verschae. Closing the gap for makespan scheduling via sparsification techniques. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 72:1–72:13, 2016. doi:10.4230/LIPIcs.ICALP.2016.72.

21 Klaus Jansen, Marten Maack, and Malin Rau. Approximation schemes for machine scheduling with resource (in-)dependent processing times. *ACM Trans. Algorithms*, 15(3):31:1–31:28, 2019. doi:10.1145/3302250.

22 Klaus Jansen and Lorant Porkolab. Linear-time approximation schemes for scheduling malleable parallel tasks. *Algorithmica*, 32(3):507–520, 2002. doi:10.1007/s00453-001-0085-8.

23 Klaus Jansen and Malin Rau. Improved approximation for two dimensional strip packing with polynomial bounded width. *Theor. Comput. Sci.*, 789:34–49, 2019. doi:10.1016/j.tcs.2019.04.002.

24 Klaus Jansen and Malin Rau. Linear time algorithms for multiple cluster scheduling and multiple strip packing. *CoRR*, abs/1902.03428, 2019. arXiv:1902.03428.

25 Klaus Jansen and Roberto Solis-Oba. Rectangle packing with one-dimensional resource augmentation. *Discrete Optimization*, 6(3):310–323, 2009. doi:10.1016/j.disopt.2009.04.001.

26 Klaus Jansen and Ralf Thöle. Approximation algorithms for scheduling parallel jobs. *SIAM Journal on Computing*, 39(8):3571–3615, 2010. doi:10.1137/080736491.

27 Berit Johannes. Scheduling parallel jobs to minimize the makespan. *Journal of Scheduling*, 9(5):433–452, 2006. doi:10.1007/s10951-006-8497-6.

28 Hans Kellerer. An approximation algorithm for identical parallel machine scheduling with resource dependent processing times. *Operations Research Letters*, 36(2):157–159, 2008.

29 Claire Kenyon and Eric Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25(4):645–656, 2000. doi:10.1287/moor.25.4.645.12118.

30 Giorgi Nadiradze and Andreas Wiese. On approximating strip packing with a better ratio than 3/2. In *27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1491–1510, 2016. doi:10.1137/1.9781611974331.ch102.

31 Martin Niemeier and Andreas Wiese. Scheduling with an orthogonal resource constraint. *Algorithmica*, 71(4):837–858, 2015. doi:10.1007/s00453-013-9829-5.

32 Ingo Schiermeyer. Reverse-fit: A 2-optimal algorithm for packing rectangles. In *2nd Annual European Symposium on Algorithms (ESA) - Algorithms*, pages 290–299, 1994. doi:10.1007/BFb0049416.

33 Daniel Dominic Sleator. A 2.5 times optimal algorithm for packing in two dimensions. *Information Processing Letters*, 10(1):37–40, 1980. doi:10.1016/0020-0190(80)90121-0.

34 A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26(2):401–409, 1997. doi:10.1137/S0097539793255801.

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

36 John Turek, Joel L. Wolf, and Philip S. Yu. Approximate algorithms scheduling parallelizable tasks. In *4th annual ACM symposium on Parallel algorithms and architectures (SPAA)*, pages 323–332, 1992. doi:10.1145/140901.141909.