Scheduling with Non-renewable Resources: Minimizing the Sum of Completion Times

Kristóf Bérczi[1], Tamás Király[1(✉)], and Simon Omlor[2]

[1] MTA-ELTE Egerváry Research Group, Department of Operations Research, Eötvös Loránd University, Budapest, Hungary
{berkri,tkiraly}@cs.elte.hu

[2] Institute for Algorithms and Complexity, TU Hamburg, Hamburg, Germany
simon.omlor@tuhh.de

Abstract. We consider single-machine scheduling problems with a non-renewable resource. In this setting, there are n jobs, each characterized by a processing time, a weight, and a resource requirement. At given points in time, certain amounts of the resource are made available to be consumed by the jobs. The goal is to assign the jobs non-preemptively to time slots on the machine, so that each job has the required resource amount available at the start of its processing. We consider the objective of minimizing the weighted sum of completion times.

The main contribution of the paper is a PTAS for the case of 0 processing times ($1|rm = 1, p_j = 0|\sum w_j C_j$). In addition, we show strong NP-hardness of the case of unit resource requirements and weights ($1|rm = 1, a_j = 1|\sum C_j$), thus answering an open question of Györgyi and Kis. We also prove that the schedule corresponding to the Shortest Processing Time First ordering provides a 3/2-approximation for the latter problem.

Keywords: Scheduling · Non-renewable resources · PTAS · Approximation algorithm

1 Introduction

Scheduling problems with non-renewable resource constraints arise naturally in various areas where resources like raw materials, energy, or funding arrive at predetermined dates. In the general setting, we are given a set of jobs and a set of machines. Each job is equipped with a requirement vector that encodes the needs of the given job for the different types of resources. There is an initial stock for each resource, and some additional resource arrival times in the future are known together with the arriving quantities. The aim is to find a schedule of the jobs on the machines such that the resource requirements are met.

Supported by DAAD with funds of the Bundesministerium für Bildung und Forschung (BMBF) and by DFG project MN 59/4-1.

M. Baïou et al. (Eds.): ISCO 2020, LNCS 12176, pp. 167–178, 2020.
https://doi.org/10.1007/978-3-030-53262-8_14

We will use the standard $\alpha|\beta|\gamma$ notation of Graham et al. [5]. Grigoriev et al. [6] extended this notation by adding the restriction $rm = r$ to the β field, meaning that there are r resources (raw materials). In the present paper, we concentrate on problem $1|rm = 1|\sum w_j C_j$, that is, when we have a single machine, a single resource, and the goal is to minimize the weighted sum of completion times.

Previous Work. Scheduling problems with resource restrictions (sometimes called financial constraints) were introduced by Carlier [2] and Slowinski [16]. Carlier settled the computational complexity of several variants for the single machine case [2]. In particular, it was shown that $1|rm = 1|\sum w_j C_j$ is NP-hard in the strong sense. This was also proved independently by Gafarov, Lazarev and Wener in [3]. Kis [15] showed that the problem remains weakly NP-hard even when the number of resource arrival times is 2 and gave an FPTAS for $1|rm = 1, q = 2|\sum w_j C_j$. A further variant of the problem was considered in [3]. Recently, Györgyi and Kis [13,14] gave polynomial time algorithms for several special cases, and also showed that the problem remains weakly NP-hard even under the very strong assumption that the processing time, the resource requirement and the weight are the same for each job. They also provided a 2-approximation algorithm for this variant. For a constant number of resource arrival times, they gave a PTAS when the processing time equals the weight for each job, and an FPTAS when the resource requirements and weights are 1.[1]

In comparison, much more is known about the maximum makespan and maximum lateness objectives. Slowinski [16] studied the preemptive scheduling of independent jobs on parallel unrelated machines with the use of additional renewable and non-renewable resources under financial constraints. Toker et al. [17] examined a single-machine scheduling problem under non-renewable resource constraint using the makespan as a performance criterion. Xie [18] generalized this result to the problem with multiple financial resource constraints. Grigoriev et al. [6] presented polynomial time algorithms, approximations and complexity results for single-machine scheduling problems with unit or all-equal processing times. In a series of papers [7–11], Györgyi and Kis presented approximation schemes and inapproximability results both for single and parallel machine problems with the makespan and the maximum lateness objectives. In [12], they proposed a branch-and-cut algorithm for minimizing the maximum lateness.

Our Results. The first problem that we consider is $1|rm = 1, a_j = 1|\sum C_j$. The complexity of this problem was posed as an open question in [12]. We show that the problem is NP-hard in the strong sense.

Theorem 1. *$1|rm = 1, a_j = 1|\sum C_j$ is strongly NP-hard.*

Given any scheduling problem on a single machine, the *Shortest Processing Time First* (SPT) schedule orders the jobs by processing times, i.e. $p_{\mathsf{spt}^{-1}(i)} \leq$

[1] Just before the submission of the present paper, Györgyi and Kis published an updated version of [14] with some new results. None of our results are implied by their paper.

$p_{\mathsf{spt}^{-1}(i+1)}$ for all i. We prove that spt provides a 3/2-approximation. We remark that it remains open whether the problem is APX-hard.

Theorem 2. *The SPT schedule gives a $\frac{3}{2}$-approximation for $1|rm = 1, a_j = 1|\sum C_j$, and the approximation guarantee is tight.*

The second problem considered is the special case when the processing time is 0 for every job. This setting is relevant to situations where processing times are negligible compared to the gaps between resource arrival times, and the bottleneck is resource availability. Examples include financial scheduling problems where the jobs are not time consuming but the availability of funding varies in time, or production problems where products are shipped at fixed time intervals and production time is negligible compared to these intervals. Note that the number of machines is irrelevant if processing times are 0.

First we present a PTAS for constant number of resource arrival times. This procedure will be used as a subroutine in our algorithm for the general case.

Theorem 3. *There exists a $(1+\frac{q}{k})$-approximation for $1|rm = 1, p_j = 0|\sum C_j w_j$ with running time $\mathcal{O}(n^{qk+1})$.*

The main contribution of the paper is a PTAS for the same problem with an arbitrary number of resource arrival times.

Theorem 4. *There exists a PTAS for $1|rm = 1, p_j = 0|\sum C_j w_j$.*

A peculiarity of the algorithm is that the PTAS for constant number of arrival times is called repeatedly on overlapping time windows, and at each call we fix only a portion of the scheduled jobs.

Due to space constraints, several proofs and details as well as further results are deferred to the full version of this paper, which is available at http://arxiv.org/abs/1911.12138.

2 Preliminaries

Throughout the paper, we will use the following notation. We are given a set J of n *jobs*. Each job $j \in J$ has a non-negative integer processing time p_j, a non-negative weight w_j, and a resource requirement a_j. The *resources arrive at time points* $t_1, \ldots, t_q$, and the *amount of resource that arrives at* t_i is denoted by b_i. We might assume that $\sum_{i=1}^{q} b_i = \sum_{j=1}^{n} a_j$ holds. We will always assume that $t_1 = 0$, as this does not effect the approximation ratio of our algorithms.

The jobs should be processed non-preemptively on a single machine. A *schedule* is an ordering of the jobs, that is, a mapping $\sigma : J \to [n]$, where $\sigma(j) = i$ means that job j is the ith job scheduled on the machine. The *completion time* of job j in schedule σ is denoted by C_j^σ. We will drop the index σ if the schedule is clear from the context. In any reasonable schedule, there is an idle time before a job j only if there is not enough resource left to start j after finishing the last job before the idle period. Hence, the completion time of job j is determined by the ordering and by the resource arrival times, as j will be scheduled at the earliest moment when the preceding jobs are already finished and the amount of available resource is at least a_j.

3 The Problem $1|rm = 1, a_j = 1| \sum C_j$

3.1 Strong NP-completeness

Proof of Theorem 1. Recall that all a_j and w_j values are 1, and each job has an integer processing time p_j. The number of resource arrival times is part of the input. We prove NP-completeness by reduction from the 3-Partition problem. The input contains numbers $B \in \mathbb{N}$, $n \in \mathbb{N}$, and $x_j \in \mathbb{N}$ $(j = 1, \ldots, 3n)$ such that $B/4 < x_j < B/2$ and $\sum_{j=1}^{3n} x_j = nB$. A feasible solution is a partition $J_1, \ldots, J_n$ of $[3n]$ such that $|J_i| = 3$ and $\sum_{j \in J_i} x_j = B$ for every $i \in [n]$. In contrast to the Partition problem, the 3-partition problem remains NP-complete even when the integers x_j are bounded above by a polynomial in n. That is, the problem remains NP-complete even when the numbers in the input are represented as unary numbers [4, pages 96–105 and 224].

Let $K = 4nB$. The reduction to $1|rm = 1, a_j = 1| \sum C_j$ involves three types of jobs. *Normal jobs* correspond to the numbers x_j in the 3-Partition instance, so there are $3n$ of them and the processing time p_j of the j-th normal job is x_j. We further have nK *small jobs* with processing time 1, and nK *large jobs* with processing time K. There are also three types of resource arrivals. The *Type 1* resource arrival times are $i(B + K)$ $(i = 0, \ldots, n - 1)$ with three resources arriving at each. The *Type 2* arrival times are $i(B + K) + j$ $(i = 0, \ldots, n - 1,$ $j = B, \ldots, B + K - 1)$ with one resource arriving. Finally, the *Type 3* resource arrival times are $n(B + K) + iK$ $(i = 0, \ldots, nK - 1)$ with one resource arriving.

Suppose that the 3-Partition instance has a feasible solution $J_1, \ldots, J_n$. We consider the following schedule S: resources of Type 1 are used by normal jobs, such that jobs in J_i are scheduled between $(i-1)(B+K)$ and $iB+(i-1)K$ (in spt order). Type 2 resources are used by small jobs that start immediately. Type 3 resources are used by the large jobs that also start immediately at the resource arrival times (see Fig. 1).

Instead of $\sum C_j$, we consider the equivalent shifted objective function $\sum(C_j - t_j - p_j)$, where t_j is the arrival time of the resource used by job j and p_j is its processing time – we assume without loss of generality that resources are used by jobs in order of arrival. Note that all terms of $\sum(C_j - t_j - p_j)$ are nonnegative. As small jobs and large jobs start immediately at the arrival of the corresponding resource in schedule S, their contribution to the shifted objective function is 0. The jobs in J_i have total processing time B, and their contribution to the shifted objective function is two times the processing time of the shortest job plus the processing time of the second shortest job, which is at most B. Hence the schedule S has objective value at most nB.

We claim that if the 3-Partition instance has no feasible solution, then the objective value of any schedule is strictly larger than nB. First, notice that if a large job is scheduled to start before time $n(B + K)$, then $\sum(C_j - t_j - p_j)$ has a term strictly larger than nB as there is a resource that arrives while the large job is processed and is not used for more than nB time units. Similarly, if the first large job starts at $n(B + K)$ but uses a resource that arrived earlier, then the resource that arrives at $n(B + K)$ is not used for more than nB time units.

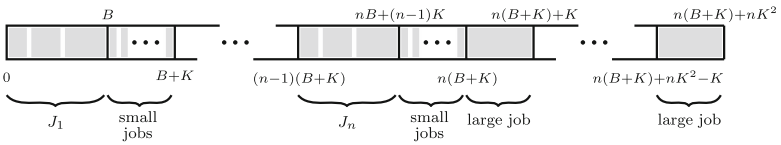

Fig. 1. The schedule corresponding to a feasible solution of 3-PARTITION.

We can conclude that the first large job uses the resource arriving at $n(B+K)$. If the first large job does not start at $n(B+K)$, then all large jobs have positive contribution to the objective value, so again, the objective value is larger than nB. We can therefore assume that the large jobs start exactly at $n(B+K)+iK$ $(i=0,\ldots,nK-1)$ and that there is no idle time before $(B+K)n$. In particular, this means that all other jobs are already completed at time $(B+K)n$.

Consider Type 2 resources arriving at $i(B+K)+j$ $(j=B,\ldots,B+K-1)$ for some fixed i. If the first or the second resource is not used immediately, then none of the subsequent ones are, so the objective value is more than nB. Hence, the first resource must be used immediately by a small job.

Suppose that some resource in this interval is used by a normal job. If it is followed by a small job, then we may improve the objective value by exchanging the two. Thus, in this case, we can assume that the last resource of the interval is used by a normal job, and also the Type 1 resources arriving at $(i+1)(B+K)$ are used by normal jobs. But this is impossible, because normal jobs have processing time at least $B/4+1$, and a small job starts at time $(i+1)(B+K)+B$.

To sum up, we can assume that all resources of Type 2 are used immediately by small jobs. This means that normal jobs have to use resources of Type 1, and must exactly fill the gaps of length B between the arrival of resources of Type 2. This is only possible if the 3-partition instance has a feasible solution, concluding the proof of Theorem 1. □

3.2 Shortest Processing Time First for Unit Resource Requirements

We now show that scheduling the jobs according to an spt ordering provides a 3/2-approximation for the problem with unit weights and unit resource requirements, thus proving Theorem 2.

Proof of Theorem 2. To prove the theorem consider any instance I. We denote the completion times for the spt ordering by C_j and their sum by spt. Furthermore, let $S_{\mathsf{spt}^{-1}(i)} := C_{\mathsf{spt}^{-1}(i)} - p_{\mathsf{spt}^{-1}(i)}$ denote the starting time of the ith job in the spt schedule. Let opt be the optimal schedule for I. We denote the completion times for opt by C'_j and their sum by opt. Let $S'_{\mathsf{opt}^{-1}(i)} := C'_{\mathsf{opt}^{-1}(i)} - p_{\mathsf{opt}^{-1}(i)}$ denote the starting time of the ith job in the optimal schedule.

Our strategy is to simplify the instance by revealing its structural properties while not decreasing $\frac{\mathsf{spt}}{\mathsf{opt}}$. We only state the claims needed for the proof here; their proofs can be found in the full version of the paper [1]. First we modify the resource arrival times.

Claim 1. *We may assume that the ith resource arrives at* $S'_{\mathsf{opt}^{-1}(i)}$ *for* $i = 1, \ldots, n$*, and that there is no idle time in schedule* opt*, that is,* $S'_{\mathsf{opt}^{-1}(i)} = C'_{\mathsf{opt}^{-1}(i-1)}$ *for* $i = 2, \ldots, n$*.*

Next, we modify the instance to have $0-1$ processing times.

Claim 2. *We may assume that* $p_{\mathsf{opt}^{-1}(1)} > p_{\mathsf{spt}^{-1}(1)}$ *and that* $p_{\mathsf{spt}^{-1}(1)} = 0$*. Furthermore, we may assume that* $p_j \in \{0, 1\}$ *for all* $j \in J$*.*

Finally, we modify the order of the jobs in the optimal solution. If opt and spt process a job of length 0 at the same time, then we can remove the job from the instance and reduce the number of resources that arrive at this time by 1. This will reduce opt and spt by the same amount.

Let t be the time at which schedule spt first starts to process a job of length 1. On one hand, opt does not process jobs of length 0 before t by the above argument. On the other hand, there is no idle time after t in spt, because that would mean idle time in opt. Thus, if we move all jobs of length 0 and their corresponding resource arrivals in opt to time t, then spt does not change but opt decreases. We may thus assume that schedule opt processes every job of length 0 at t.

We conclude that opt first processes k_1 jobs of length 1, then k_1 jobs of length 0 and then k_2 jobs of length 1, while spt starts with the jobs of length 0 having a lot of idle time in the beginning and then consecutively processes all jobs of length 1. The weighted sums of completion times are then given by $\mathrm{opt} = \frac{k_1(k_1+1)}{2} + k_1^2 + k_2 k_1 + \frac{k_2(k_2+1)}{2}$ and $\mathrm{spt} = \frac{k_1(k_1-1)}{2} + k_2 k_1 + \frac{k_1(k_1+1)}{2} + (k_1 + k_2)k_1 + \frac{k_2(k_2+1)}{2}$. We get $\frac{3}{2}\mathrm{opt} - \mathrm{spt} = \frac{k_1^2}{4} + \frac{k_2^2}{4} - \frac{k_1 k_2}{2} + \frac{3k_1+k_2}{4} \geq \frac{(k_1-k_2)^2}{4} \geq 0$, showing that the approximation factor is at most $\frac{3}{2}$.

Setting $k_2 = k_1$ and letting k_1 go to infinity gives us a sequence of instances such that $\frac{\mathrm{spt}}{\mathrm{opt}}$ converges to $\frac{3}{2}$ as we have $\mathrm{spt} = \frac{9}{2}k_1^2 + \mathcal{O}(k_1)$ and $\mathrm{opt} = 3k_1^2 + \mathcal{O}(k_1)$. This concludes the proof of Theorem 2. □

4 The Problem $1|rm = 1, p_j = 0|\sum C_j w_j$

In this section we consider problem $1|rm = 1, p_j = 0|\sum C_j w_j$, another special case of $1|rm = 1|\sum C_j w_j$. Since the processing times are 0, every job is processed at one of the arrival times in any optimal schedule. Thus, a schedule can be represented by a mapping $\pi : J \to [q]$, where $\pi(j)$ denotes the index of the resource arrival time when job j is processed. A schedule is feasible if the resource requirements are met, that is, if $\sum_{j:\pi(j)\leq k} a_j \leq \sum_{i \leq k} b_i$ for all $1 \leq k \leq q$. As we assume that $\sum_i b_i = \sum_j a_j$ holds, this is equivalent to $\sum_{j:\pi(j)\geq k} a_j \geq \sum_{i \geq k} b_i$ for all $1 \leq k \leq q$.

Define $B_k = \sum_{i \geq k} b_i$, and consider the set of jobs that are not processed before a given time point t_i. Then the second inequality says that if the resource requirements of these jobs add up to at least B_i, then our schedule is feasible.

4.1 PTAS for Constant q

The aim of this section is to give a PTAS for the case when the number of resource arrival times is a constant. The algorithm is a generalization of a well known PTAS for the knapsack problem, and will be used later as a subroutine in the PTAS for an arbitrary number of resource arrival times. The idea is to choose a number $k \in \mathbb{Z}_+$, guess the k heaviest jobs that are processed at each resource arrival time t_i, and then determine the remaining jobs that are scheduled at t_i in a greedy manner. Since we go over all possible sets containing at most k jobs for each resource arrival time, there is an exponential dependence on the number q of resource arrival times in the running time.

Algorithm 1. PTAS for $1|rm = 1, p_j = 0| \sum C_j w_j$ when q is a constant.

Input: Jobs J with $|J| = n$, resource requirements a_j, weights w_j, resource arrival times $t_1 \leq \ldots \leq t_q$ and resource quantities $b_1, \ldots b_q$.
Output: A feasible schedule π.

1: **for all** subpartitions $S_1 \cup \cdots \cup S_q \subseteq J$ with $|S_i| \leq k$ for $i > 1$ **do**
2: Set $A = 0$.
3: Set $W = 0$.
4: **for** i from 0 to $q - 2$ **do**
5: **for** $j \in S_{q-i}$ **do**
6: $\pi(j) = q - i$
7: $A \leftarrow A + a_j$
8: **if** $|S_{q-i}| = k$ **then**
9: $W \leftarrow \max\{W, \min\{w_j : \ j \in S_{q-i}\}\}$
10: **while** $A < B_{q-i}$ **do**
11: **if** there exists an unassigned job j with $w_j \leq W$ **then**
12: Let j be an unassigned job with $w_j \leq W$ minimizing w_j/a_j.
13: $\pi(j) = q - i$
14: $A \leftarrow A + a_j$
15: **else**
16: **break**
17: For all remaining jobs set $\pi(j) = 1$.
18: Let π be the best schedule found.
19: **return** π

Proof of Theorem 3. We claim that Algorithm 1 satisfies the requirements of the theorem. Let π^{opt} be an optimal schedule and define $J_i^{\mathsf{opt}} = \{j \in J : \ \pi^{\mathsf{opt}}(j) = i\}$. Let S_i^{opt} be the set of the k heaviest jobs in J_i^{opt} if $|J_i^{\mathsf{opt}}| \geq k$, otherwise let $S_i^{\mathsf{opt}} = J_i^{\mathsf{opt}}$. Let $J_i = \{j \in J : \ \pi(j) = i\}$ denote the set of jobs assigned to time t_i in our solution. In each iteration of the *for* loop of Step 4, let j_i be the last job added to J_i if such a job exists.

Assume that we are at the iteration of the algorithm when the subpartition $S_1^{\mathsf{opt}} \cup \cdots \cup S_q^{\mathsf{opt}}$ is considered in Step 1. Let $W_{q-\ell}$ denote the value of W at the end of the iteration of the *for* loop corresponding to $i = \ell$ in Step 4. By Steps 3

and 9, we have $W_{q-\ell} \leq \frac{1}{k} \sum_{i=\ell}^{q} \sum_{j \in J_i^{\mathsf{opt}}} w_j$. As our algorithm always picks the most inefficient job, we also have $\sum_{i=\ell}^{q} \sum_{j \in J_i \setminus \{j_i\}} w_j \leq \sum_{i=\ell}^{q} \sum_{j \in J_i^{\mathsf{opt}}} w_j$, where $J_i \setminus \{j_i\} = J_i$ if j_i is not defined for i.

Combining these two observations, for $\ell = 1, \ldots, q$ we get

$$\sum_{i=\ell}^{q} \sum_{j \in J_i} w_j = \sum_{i=\ell}^{q} \sum_{j \in J_i \setminus \{j_i\}} w_j + \sum_{i=\ell}^{q} w_{j_i}$$
$$\leq \sum_{i=\ell}^{q} \sum_{j \in J_i^{\mathsf{opt}}} w_j + (q - \ell + 1) \cdot W_\ell \leq (1 + \frac{q}{k}) \sum_{i=\ell}^{q} \sum_{j \in J_i^{\mathsf{opt}}} w_j,$$

where the first inequality follows from the fact that $w_{j_i} \leq W_i \leq W_\ell$ whenever $i \geq \ell$. This proves that the schedule that we get is a $(1 + \frac{q}{k})$-approximation.

We get a factor of n^{qk} in the running time for guessing the sets S_k. Assigning the remaining jobs can be done in linear time by ordering the jobs and using AVL-trees, thus we get an additional factor of n. In order to get a PTAS, we set $k = \frac{\varepsilon}{q}$, concluding the proof of the theorem. □

4.2 PTAS for Arbitrary q

We turn to the proof of the main result of the paper. The first idea is to shift resource arrival times to powers of $1 + \varepsilon$, for a suitably small ε.

Let $\mathcal{I}$ be an instance of $1|rm = 1, p_j = 0| \sum_j C_j w_j$. We assume that resource arrival times are integer, and that $t_1 = 0$, $t_2 = 1$. We define a new instance $\mathcal{I}'$ of $1|rm = 1, p_j = 0| \sum_j C_j w_j$ with shifted resource arrival times as follows. Set $t'_1 = 0$ and $t'_i = (1+\varepsilon)^{i-2}$ for $i = 2, \ldots, \lceil \log_{1+\varepsilon}(t_q) \rceil + 2$. Moreover, let $b'_1 = b_1, b'_2 = b_2$ and $b'_i = \sum [b_i : \ t_i \in ((1+\varepsilon)^{i-3}, (1+\varepsilon)^{i-2}]$ for $i = 3, \ldots, \lceil \log_{1+\varepsilon}(t_q) \rceil + 2$.

The proof of the following claim is the same as that of Claim 12 in [1].

Claim 3. *A solution to $\mathcal{I}$ with weighted sum of completion times W can be transformed into a solution of $\mathcal{I}'$ with weighted sum of completion times at most $(1+\varepsilon)W$. Furthermore, any feasible schedule for $\mathcal{I}'$ is also feasible for $\mathcal{I}$.* □

Due to the claim, we may assume that the positive arrival times are powers of $1 + \varepsilon$. For convenience of notation, we will assume in this subsection that the largest arrival time is 1, and arrival times are indexed in decreasing order, starting with $t_0 = 1$. That is, $t_i = (1 + \varepsilon)^{-i}$ $(i = 0, \ldots, q - 2)$, and $t_{q-1} = 0$. We will also assume that for a given constant r, $b_{q-r-1} = \cdots = b_{q-2} = 0$. This can be achieved by adding r dummy arrival times.

Proof of Theorem 4. Let us fix an even integer r and $\varepsilon > 0$; we will later assume that r is very large compared to ε^{-1}. We assume that resource arrival times are as described above. The algorithm is given as Algorithm 2.

In the algorithm, we fix jobs at progressively decreasing arrival times, by using the PTAS of the previous section for $r + 1$ arrival times on different instances except for the first step, when we may use the PTAS for less than $r+1$

Algorithm 2. PTAS for $1|rm = 1, p_j = 0| \sum C_j w_j$

Input: Jobs J with $|J| = n$, resource requirements a_j, weights w_j; an even integer r; resource quantities $b_0, \dots b_{q-1}$ such that $b_{q-r-1} = \cdots = b_{q-2} = 0$ and $\sum a_j = \sum b_i$. We assume resource arrival times are $t_i = (1+\varepsilon)^{-i}$ $(i = 0, \dots, q-2)$, $t_{q-1} = 0$.
Output: A feasible schedule π.

for ℓ from 1 to $r/2$ **do**
  Obtain instance $\mathcal{I}'$ with $r/2+\ell+1$ arrival times by moving arrivals before $t_{r/2+\ell-1}$ to 0.
  Run Algorithm 1 on $\mathcal{I}'$ to get schedule σ.
  Let $A = B = 0$.
  for i from 0 to $\ell - 1$ **do**
    For every $j \in \sigma^{-1}(i)$, fix $\pi_\ell(j) = i$.
    $A \leftarrow A + \sum_{j \in \sigma^{-1}(i)} a_j$
    $B \leftarrow B + b_i$
  for j from 2 to $\lfloor 2(q-1-\ell)/r \rfloor$ **do**
    Let $s = (j-2)r/2 + \ell$.
    Obtain instance $\mathcal{I}'$ with arrival times $t_s, t_{s+1}, \dots, t_{s+r-1}, 0$: remove arrivals after t_s, remove $\max\{A-B, 0\}$ latest remaining resources, and move all arrivals before t_{s+r-1} to 0.
    Let $A = B = 0$.
    Run Algorithm 1 on $\mathcal{I}'$ to get schedule σ.
    for i from s to $s + r/2 - 1$ **do**
      For every $j \in \sigma^{-1}(i)$, fix $\pi_\ell(j) = i$.
      $A \leftarrow A + \sum_{j \in \sigma^{-1}(i)} a_j$
      $B \leftarrow B + b_i$
  For all unscheduled jobs j, set $\pi_\ell(j) = q - 1$.
Let π be the best schedule among $\pi_1, \dots, \pi_{r/2}$.
return π

arrival times. We will run our algorithm $r/2$ times with slight modifications, and pick the best result. Each run is characterized by a parameter $\ell \in \{1, \dots, r/2\}$.

In the first step, we consider arrival times $t_0, t_1, \dots, t_{r/2+\ell-1}, 0$. We move the resources arriving before $t_{r/2+\ell-1}$ to 0, and use the PTAS for $r/2 + \ell + 1$ arrival times on this instance. We fix the jobs that are scheduled at arrival times $t_0, t_1, \dots, t_{\ell-1}$.

Consider now the jth step for some $j \geq 2$. Define $s = (j-2)r/2 + \ell$ and consider arrival times $t_s, t_{s+1}, \dots, t_{s+r-1}, 0$. Move the resources arriving before t_{s+r-1} to 0, and decrease $b_s, b_{s+1}, \dots$ in this order as needed, so that the total requirement of unfixed jobs equals the total resource. Use the PTAS for $r+1$ arrival times on this instance. Fix the jobs that are scheduled at arrival times $t_s, t_{s+1}, \dots, t_{s+r/2-1}$. The algorithm runs while $s + r - 1 \leq q - 2$, i.e., $jr/2 + \ell \leq q - 1$. Since the smallest r arrival times (except for 0) are dummy arrival times, the algorithm considers all resource arrivals.

The schedule given by the algorithm is clearly feasible, because when jobs at t_i are fixed, the total resource requirement of jobs starting no earlier than t_i is at least the total amount of resource arriving no earlier than t_i. To analyze

the approximation ratio, we introduce the following notation: W_i is the total weight that the algorithm schedules at t_i; W'_i is the weight that the algorithm temporarily schedules at t_i when i is in the interval $[t_{s+r/2}, t_{s+r-1}]$ (or, in the first step, in the interval $[t_\ell, t_{\ell+r/2-1}]$); W^*_i is the total weight scheduled at t_i in the optimal solution.

Since we use the PTAS for $r/2+\ell+1$ arrival times in the first step, we have $\sum_{i=0}^{\ell-1}(1+\varepsilon)^{-i}W_i + \sum_{i=\ell}^{\ell+r/2-1}(1+\varepsilon)^{-i}W'_i \le (1+\varepsilon)\sum_{i=0}^{\ell+r/2-1}(1+\varepsilon)^{-i}W^*_i$, as the right-hand side is $(1+\varepsilon)$ times the objective value of the feasible solution obtained from the optimal solution by moving jobs arriving before $t_{\ell+r/2-1}$ to 0.

For $s = jr/2+\ell$, we compare the output of the PTAS with a different feasible solution: we schedule total weight W'_i at t_i for $i = s, s+1, \ldots, s+r/2-1$, total weight W^*_i at t_i for $i = s+r/2+1, \ldots, s+r-1$, and at $t_{s+r/2}$ we schedule all jobs that are no earlier than $t_{s+r/2}$ in the optimal schedule but are no later than $t_{s+r/2}$ in the PTAS schedule. We get the inequality

$$\sum_{i=jr/2+\ell}^{(j+1)r/2+\ell-1}(1+\varepsilon)^{-i}W_i + \sum_{i=(j+1)r/2+\ell}^{(j+2)r/2+\ell-1}(1+\varepsilon)^{-i}W'_i \le (1+\varepsilon)\left(\sum_{i=jr/2+\ell}^{(j+1)r/2+\ell-1}(1+\varepsilon)^{-i}W'_i + \sum_{i=(j+1)r/2+\ell}^{(j+2)r/2+\ell-1}(1+\varepsilon)^{-i}W^*_i + (1+\varepsilon)^{-(j+1)r/2-\ell}\sum_{i=0}^{(j+1)r/2+\ell-1}W^*_i\right).$$

The sum of these inequalities gives

$$\sum_{i=0}^{q-2}(1+\varepsilon)^{-i}W_i \le \varepsilon\sum_{i=\ell}^{q-2}(1+\varepsilon)^{-i}W'_i + (1+\varepsilon)\sum_{i=0}^{q-2}(1+\varepsilon)^{-i}W^*_i + (1+\varepsilon)\sum_{i=0}^{q-2}\left(\sum_{j:jr/2+\ell>i}(1-\varepsilon)^{-(jr/2+\ell)}\right)W^*_i. \tag{1}$$

To bound the first term on the right hand side of (1), first we observe that $\sum_{i=\ell}^{r/2+\ell-1}(1+\varepsilon)^{-i}W'_i \le (1+\varepsilon)\sum_{i=0}^{r/2+\ell-1}(1+\varepsilon)^{-i}W^*_i$, because the left side is at most the value of the PTAS in the first step, while the right side is $(1+\varepsilon)$ times the value of a feasible solution. Similarly,

$$\sum_{i=(j+1)r/2+\ell}^{(j+2)r/2+\ell-1}(1+\varepsilon)^{-i}W'_i \le (1+\varepsilon)\left(\sum_{i=jr/2+\ell}^{(j+2)r/2+\ell-1}(1+\varepsilon)^{-i}W^*_i + (1+\varepsilon)^{-jr/2-\ell}\sum_{i=0}^{jr/2+\ell-1}W^*_i\right),$$

because the left side is at most the value of the PTAS in the $(j+1)$-th step, and the right side is $(1+\varepsilon)$ times the value of the following feasible solution: take the optimal solution, move jobs scheduled before $t_{(j+2)r/2+\ell-1}$ to 0, and move

jobs scheduled after $t_{jr/2+\ell}$ to $t_{jr/2+\ell}$. Adding these inequalities, we get

$$\begin{aligned}
&\varepsilon \sum_{i=\ell}^{q-2} (1+\varepsilon)^{-i} W_i' \\
&\leq \varepsilon(1+\varepsilon)\left(2\sum_{i=0}^{q-2}(1+\varepsilon)^{-i}W_i^* + \sum_{i=0}^{q-2}\left(\sum_{j:jr/2+\ell>i}(1+\varepsilon)^{-jr/2-\ell}\right)W_i^*\right) \\
&\leq \varepsilon\left(2(1+\varepsilon) + \frac{(1+\varepsilon)^{r/2}}{(1+\varepsilon)^{r/2}-1}\right)\sum_{i=0}^{q-2}(1+\varepsilon)^{-i}W_i^*.
\end{aligned}$$

The last expression is at most 4ε times the optimum value if r is large enough.

The last term of the right side of (1) is too large to get a bound that proves a PTAS. However, we can bound the *average* of these terms for different values of ℓ. The average is

$$\begin{aligned}
&(1+\varepsilon)\frac{2}{r}\sum_{\ell=1}^{r/2}\sum_{i=0}^{q-2}\left(\sum_{j:jr/2+\ell>i}(1-\varepsilon)^{-(jr/2+\ell)}\right)W_i^* \\
&\leq (1+\varepsilon)\frac{2}{r}\sum_{i=0}^{q-2}\left(\sum_{j=1}^{\infty}(1+\varepsilon)^{-j}\right)(1-\varepsilon)^{-i}W_i^* = (1+\varepsilon)\frac{2}{r\varepsilon}\sum_{i=0}^{q-2}(1-\varepsilon)^{-i}W_i^*,
\end{aligned}$$

which is at most ε times the optimum if r is large enough. To summarize, we obtained that for large enough r, the average objective value of our algorithm for $\ell = 1, 2, \ldots, r/2$ is upper bounded by

$$4\varepsilon\sum_{i=0}^{q-2}(1+\varepsilon)^{-i}W_i^* + (1+\varepsilon)\sum_{i=0}^{q-2}(1+\varepsilon)^{-i}W_i^* + \varepsilon\sum_{i=0}^{q-2}(1+\varepsilon)^{-i}W_i^* = (1+6\varepsilon)\sum_{i=0}^{q-2}(1+\varepsilon)^{-i}W_i^*,$$

which is $(1+6\varepsilon)$ times the objective value of the optimal solution. This proves that the algorithm that chooses the best of the $r/2$ runs is a PTAS. □

Acknowledgement. The authors are grateful to Erika Bérczi-Kovács and to Matthias Mnich for the helpful discussions. Kristóf Bérczi was supported by the János Bolyai Research Fellowship of the Hungarian Academy of Sciences and by the ÚNKP-19-4 New National Excellence Program of the Ministry for Innovation and Technology. Projects no. NKFI-128673 and ED18-1-2019-0030 have been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the FK_18 and Thematic Excellence Programme funding schemes. Tamás Király was supported by NKFIH grant number K120254.

References

1. Bérczi, K., Király, T., Omlor, S.: Scheduling with non-renewable resources: minimizing the sum of completion times (2019). https://arxiv.org/abs/1911.12138. Preprint
2. Carlier, J.: Problèmes d'ordonnancement à contraintes de ressources: algorithmes et complexité. Université Paris VI-Pierre et Marie Curie, Institut de programmation (1984)

3. Gafarov, E.R., Lazarev, A.A., Werner, F.: Single machine scheduling problems with financial resource constraints: some complexity results and properties. Math. Soc. Sci. **62**(1), 7–13 (2011)
4. Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NP-Completeness (1979)
5. Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Annals of discrete mathematics, vol. 5, pp. 287–326. Elsevier (1979)
6. Grigoriev, A., Holthuijsen, M., Van De Klundert, J.: Basic scheduling problems with raw material constraints. Naval Res. Logist. (NRL) **52**(6), 527–535 (2005)
7. Györgyi, P.: A ptas for a resource scheduling problem with arbitrary number of parallel machines. Oper. Res. Lett. **45**(6), 604–609 (2017)
8. Györgyi, P., Kis, T.: Approximation schemes for single machine scheduling with non-renewable resource constraints. J. Sched. **17**(2), 135–144 (2013). https://doi.org/10.1007/s10951-013-0346-9
9. Györgyi, P., Kis, T.: Approximability of scheduling problems with resource consuming jobs. Ann. Oper. Res. **235**(1), 319–336 (2015). https://doi.org/10.1007/s10479-015-1993-3
10. Györgyi, P., Kis, T.: Reductions between scheduling problems with non-renewable resources and knapsack problems. Theoret. Comput. Sci. **565**, 63–76 (2015)
11. Györgyi, P., Kis, T.: Approximation schemes for parallel machine scheduling with non-renewable resources. Eur. J. Oper. Res. **258**(1), 113–123 (2017)
12. Györgyi, P., Kis, T.: Minimizing the maximum lateness on a single machine with raw material constraints by branch-and-cut. Comput. Ind. Eng. **115**, 220–225 (2018)
13. Györgyi, P., Kis, T.: Minimizing total weighted completion time on a single machine subject to non-renewable resource constraints. J. Sched. **22**(6), 623–634 (2019). https://doi.org/10.1007/s10951-019-00601-1
14. Györgyi, P., Kis, T.: New complexity and approximability results for minimizing the total weighted completion time on a single machine subject to non-renewable resource constraints. arXiv preprint arXiv:2004.00972 (2020). Earlier version: EGRES Technical Report 2019–05, egres.elte.hu
15. Kis, T.: Approximability of total weighted completion time with resource consuming jobs. Oper. Res. Lett. **43**(6), 595–598 (2015)
16. Slowiński, R.: Preemptive scheduling of independent jobs on parallel machines subject to financial constraints. Eur. J. Oper. Res. **15**(3), 366–373 (1984)
17. Toker, A., Kondakci, S., Erkip, N.: Scheduling under a non-renewable resource constraint. J. Oper. Res. Soc. **42**(9), 811–814 (1991)
18. Xie, J.: Polynomial algorithms for single machine scheduling problems with financial constraints. Oper. Res. Lett. **21**(1), 39–42 (1997)