Journal of Scheduling (2024) 27:151–164
https://doi.org/10.1007/s10951-024-00807-y

Scheduling with non-renewable resources: minimizing the sum of completion times

Kristóf Bérczi[1] · **Tamás Király**[2] · **Simon Omlor**[3]

Accepted: 5 February 2024 / Published online: 24 March 2024

Abstract

We consider single-machine scheduling with a non-renewable resource. In this setting, we are given a set of jobs, each characterized by a processing time, a weight, and a resource requirement. At fixed 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 enough resource available at the start of its processing. The objective that we consider is the minimization of the sum of weighted 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. Finally, we investigate a variant of the problem where processing times are 0 and the resource arrival times are unknown. We present a $(4+\epsilon)$-approximation algorithm, together with a $(4-\varepsilon)$-inapproximability result, for any $\varepsilon > 0$.

Keywords Approximation algorithm · Non-renewable resources · Polynomial-time approximation scheme · Strong NP-hardness · Scheduling · Weighted sum of completion times

1 Introduction

Scheduling problems with non-renewable resource constraints arise naturally in various areas where resources like raw materials, energy, or financial 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 necessary resources are available for each job when their processing begins.

We will use the standard $\alpha|\beta|\gamma$ notation of Graham et al. (1979). Grigoriev et al. (2005) extended this notation by adding the restriction $rm = r$ to the β field, meaning that there are r resources (rm stands for 'raw materials'). In the present paper, we concentrate on problem with a single machine and a single resource, where the objective is to minimize the weighted sum of completion times, i.e., $1|rm = 1|\sum w_j C_j$. While there is an abundance of results on the approximability of the makespan objective, much less

An extended abstract of this work appeared in the 6th International Symposium on Combinatorial Optimization (ISCO 2020). This research has been implemented with the support provided by the Lendület Programme of the Hungarian Academy of Sciences—Grant Number LP2021-1/2021, by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the ELTE TKP 2021-NKTA-62 funding scheme, and by Dynasnet European Research Council Synergy project (ERC-2018-SYG 810115).

✉ Kristóf Bérczi
kristof.berczi@ttk.elte.hu

Tamás Király
tamas.kiraly@ttk.elte.hu

Simon Omlor
simon.omlor@tu-dortmund.de

[1] MTA-ELTE Matroid Optimization Research Group, HUN-REN-ELTE Egerváry Research Group, Department of Operations Research, Eötvös Loránd University, Budapest, Hungary

[2] HUN-REN-ELTE Egerváry Research Group, Department of Operations Research, Eötvös Loránd University, Budapest, Hungary

[3] Faculty of Statistics, TU Dortmund University, Dortmund, Germany

is known about the complexity and approximability of the total weighted completion time objective.

Related work Scheduling problems with resource restrictions (also called financial constraints, or raw material requirements) were introduced by Carlier and Kan (1982) and by Slowiński (1984). Carlier (1984) settled the computational complexity of several variants for the single machine case. In particular, he showed that $1|rm = 1|\sum w_j C_j$ is NP-hard in the strong sense. This was also proved independently by Gafarov et al. (2011). Kis (2015) showed that the problem remains weakly NP-hard even when the number of resource arrival times is 2. On the positive side, he gave an FPTAS for $1|rm = 1, q = 2|\sum w_j C_j$. A variant of the problem where each job has processing time 1, there are $q = n$ resource arrival times such that $t_i = iM$ and $b_i = M$ for $i = 1, \ldots, n$, and $M = \sum_{j \in J} a_j/n$ is an integer, was considered in Gafarov et al. (2011). Györgyi and Kis (2019) 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 for each individual job, the processing time, the resource requirement and the weight are equal. They also provided a 2-approximation algorithm for this variant, and a polynomial-time approximation scheme (PTAS) for the variant where the number of resource arrival times is a constant and the processing time equals the weight for each job, while the resource requirements are arbitrary.

Independently of the present paper, Györgyi and Kis (2020) recently published an analysis of simple greedy algorithms for several variants of the problem. They also showed that minimizing the sum of completion times is NP-hard even for two resource arrival times and unit resource requirements, and provided a FPTAS for a variant in which the jobs have arbitrary weights, but the number of resource arrival times is bounded by a constant. None of our results are implied by their paper.

In comparison to total weighted completion time, much more is known about the maximum makespan and maximum lateness objectives. Slowiński (1984) 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. (1991) examined a single-machine scheduling problem under non-renewable resource constraint, using the makespan as a performance criterion. Xie (1997) generalized this result to the problem with multiple financial resource constraints. Grigoriev et al. (2005) presented polynomial time algorithms, approximations and complexity results for single-machine scheduling problems with unit or all-equal processing times and maximum lateness and makespan objectives. In a series of papers (Györgyi & Kis, 2014, 2015a, 2015b, 2017, Györgyi, 2017), 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 Györgyi and Kis (2018), they proposed a branch-and-cut algorithm for minimizing the maximum lateness.

Our results We first consider the problem $1|rm = 1, a_j = 1|\sum C_j$. The complexity of this problem was posed as an open question in Györgyi and Kis (2018). 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.*

In the light of Theorem 1, one might be interested in finding an approximation algorithm for the problem. Given any scheduling problem on a single machine, the **Shortest Processing Time First** (SPT) schedule orders the jobs by increasing order of processing times. We prove that spt provides a 3/2-approximation. Although the algorithm is very simple as it is merely scheduling according to the SPT order, the analysis of the approximation factor is rather involved.

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 describe a fast and simple greedy approximation algorithm for the problem.

Theorem 3 *For $1|rm = 1, p_j = 0|\sum C_j w_j$, there exists a 6-approximation algorithm with running time $\mathcal{O}(n \log n)$.*

After the proof of Theorem 3, we give a slightly more complicated $(4+\varepsilon)$-approximation that illustrates one of the important ideas of the general PTAS.

As a next step toward the main result, we present a PTAS for the case of a constant number of resource arrival times. This procedure will be used as a subroutine in our algorithm for the general case.

Theorem 4 *Consider the number of arrival times q to be constant. For any fixed positive integer k, there is a $(1+\frac{q}{k})$-approximation algorithm 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 5 *There exists a PTAS for* $1|rm = 1, p_j = 0|\sum C_j w_j$.

Finally, we consider a variant of $1|rm = 1, p_j = 0|\sum C_j w_j$ where the number of resource arrival times and the arriving quantities (in the order of the arrivals) are known, but the arrival times are unknown. We denote this problem by $1|rm = 1, p_j = 0, t_i \text{ unknown}|\sum C_j w_j$. We observe that the greedy 6-approximation algorithm of Theorem 3 is actually a 6-approximation for this problem, too. We can improve the approximation factor to get the following tight result.

Theorem 6 *For* $1|rm = 1, p_j = 0, t_i$ *unknown*$|\sum C_j w_j$, *there exists a* $(4+\varepsilon)$*-approximation with running time polynomial in* $1/\varepsilon$ *and the input length. Moreover, there is no* $(4-\varepsilon)$*-approximation algorithm for the problem for any* $\varepsilon > 0$.

Organization The rest of the paper is organized as follows. Basic notation and terminology are introduced in Sect. 2. A strong NP-hardness proof and a 3/2-approximation algorithm for problem $1|rm = 1, a_j = 1|\sum C_j$ are given in Sect. 3. Results on problem $1|rm = 1, p_j = 0|\sum C_j$ are discussed in Sect. 4, where a greedy 6-approximation, a PTAS for the case of constant resource arrival times, and a PTAS for the general case are presented. We close the paper in Sect. 5 by analyzing the variant where the resource arrival times are unknown.

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 affect the approximation ratio of our algorithms. We will use the notation $B_k = \sum_{i \geq k} b_i$ for the amount of resource that arrives no earlier than t_k.

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 first moment when the preceding jobs are already finished and the amount of available resource is at least a_j.

A different representation of schedules will be used in Sect. 4, where the processing times are assumed to be 0. In this case, every job is processed at one of the resource arrival times in any reasonable schedule. Hence, 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.

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

3.1 Strong NP-completeness

The aim of this section is to prove Theorem 1.

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

Proof 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 $\sum_{j=1}^{3n} x_j = nB$ (note that we will not use the upper bound $x_j < B/2$ in the proof). 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 (Garey & Johnson, 1979, Pages 96–105 and 224).

We assume without loss of generality that B is divisible by 4, so $x_j \geq B/4 + 1 \geq 2$ for every j. Let $K = 4nB$. The reduction to $1|rm = 1, a_j = 1|\sum C_j$ involves three types of jobs.

Normal jobs These 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 jth normal job is x_j.

Small jobs Their processing time is 1 and there are nK of them.

Large jobs Their processing time is K and there are nK of them.

There are also three types of resource arrivals (see Fig. 1):

Type 1 Three resources arrive at times $i(B+K)$ $(i = 0, \ldots, n-1)$.

Type 2 One resource arrives at $i(B+K)+j$ $(i = 0, \ldots, n-1, j = B, \ldots, B+K-1)$.

Type 3 One resource arrives at $n(B+K)+iK$ $(i = 0, \ldots, nK-1)$.

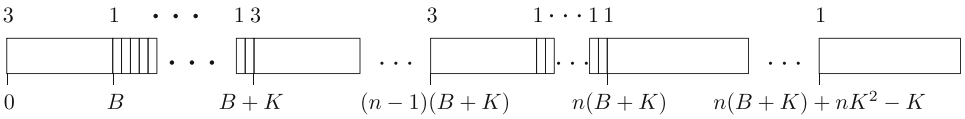

Fig. 1 Resource arrivals in the reduction of 3- PARTITION

Suppose that the 3- PARTITION instance has a feasible solution $J_1, \ldots, J_n$. We consider the following schedule σ: 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. 2).

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 the processing time of j—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 σ, 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 twice the processing time of the shortest job, plus the processing time of the second shortest job, which is at most B. Hence the schedule σ 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. 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 $i \leq n-1$. If the first resource or the second resource in this interval is not used immediately, then none of the subsequent ones are, so the objective value is at least $K - 1 > nB$. Hence, we may assume that both the first and the second resources are used immediately. This means that first resource is used immediately by a small job, since normal jobs have processing time at least 2. Thus, the resource arriving at $i(B+K)+B$ is immediately used by a small job, for every $i \leq n-1$.

Suppose that some other resource in the interval $i(B+K)+j$ $(j = B+1, \ldots, B+K-1)$ 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 (this already implies $i \leq n-2$, because a large job starts at $n(B+K)$), 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 $(i+1)(B+K)+B$ by.

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

In the previous section, we have seen that scheduling with a non-renewable resource is strongly NP-hard already for unit resource requirements. Now we show that scheduling the jobs according to an spt ordering provides a 3/2-approximation for the problem with unit weight and unit resource requirements, thus proving Theorem 2.

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.*

Proof Consider an instance I of the problem. Let σ_{spt} and σ_{opt} denote the SPT and the optimal schedule, and let spt and opt denote the sum of the completion times in these two schedules, respectively. We will use the notation $j_{\mathsf{spt}}(i)$, $p_{\mathsf{spt}}(i)$, $S_{\mathsf{spt}}(i)$, and $C_{\mathsf{spt}}(i)$ for the ith job in the SPT schedule, its processing time, its starting time, and its completion time, respectively. We also use similar notation with subscript opt for the optimal schedule.

Our strategy is to simplify the instance by revealing its structural properties while not decreasing $\frac{\mathsf{spt}}{\mathsf{opt}}$. This way we get an upper bound for the approximation factor. We first consider the resource arrival times.

Claim 1 *We may assume that the ith resource arrives at $S_{\mathsf{opt}}(i)$ for $i = 1, \ldots, n$.*

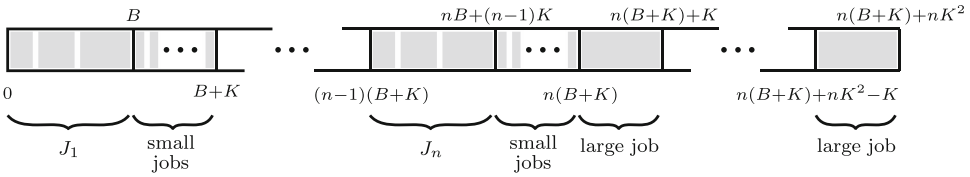

Fig. 2 The schedule corresponding to a feasible solution of 3- PARTITION

Proof As the ith resource is used by job $j_{\mathsf{opt}}(i)$, the arrival time of that resource is at most $S_{\mathsf{opt}}(i)$. If we move the arrival time of the resource to exactly $S_{\mathsf{opt}}(i)$, then opt does not change and spt cannot decrease. □

The next claim shows that we can get rid of the idle times in the optimal schedule.

Claim 2 *We may assume that there is no idle time in schedule* opt*, that is,* $S_{\mathsf{opt}}(i) = C_{\mathsf{opt}}(i-1)$ *for* $i = 2, \ldots, n$.

Proof Suppose that there is some i such that $t_i > C_{\mathsf{opt}}(i-1)$. We reduce $t_{i'}$ by $\Delta = t_i - C_{\mathsf{opt}}(i-1)$ for all $i' \geq i$. Then for each $i' \geq i$, the completion time $C_{\mathsf{opt}}(i')$ decreases by Δ. For each $i' \geq i$, the completion time $C_{\mathsf{spt}}(i')$ decreases by at most Δ. This follows from the fact that the resource arrival times decrease by Δ and the completion time of the previous job can decrease by at most Δ (which can be shown by induction). Hence opt decreases by at least as much as spt. Since spt $\geq$ opt, the ratio $\frac{\mathsf{spt}}{\mathsf{opt}}$ will not decrease by this change. □

Next, we modify the processing times.

Claim 3 *We may assume that* $p_{\mathsf{opt}}(1) > p_{\mathsf{spt}}(1)$ *and that* $p_{\mathsf{spt}}(1) = 0$.

Proof If both schedules start with the same job, then we can remove the job from the instance and decrease b_1 by 1. Then opt decreases by the same amount as spt. We can repeat this until the schedules start with jobs of different processing times. Now $p_{\mathsf{opt}}(1) > p_{\mathsf{spt}}(1)$, since spt starts with the shortest job. Decreasing the processing time of job $j_{\mathsf{spt}}(1)$ to 0 (without changing any arrival time) decreases spt by $p_{\mathsf{spt}}(1)$ and opt by at least $p_{\mathsf{spt}}(1)$. We can eliminate idle times in the new optimal schedule as in the proof of Claim 2. □

Claim 4 *We may assume that* $p_j \in \{0, 1\}$ *for all* $j \in J$.

Proof Let $p_{\max} = \max_{j \in J} p_j$ be the maximum processing time. Scaling the processing times by dividing all processing and arrival times by $p_{\max}$ has no effect on $\frac{\mathsf{spt}}{\mathsf{opt}}$, hence we may assume that $p_{\max} = 1$. Now assume that there is a job j' with $p = p_{j'} \in (0, 1)$. Let $\overline{p} = \min\{p_j \mid j \in J, p_j > p\}$ and $\underline{p} = \max\{p_j \mid j \in J, p_j < p\}$. Let $J_p = \{j \in J \mid p_j = p\}$ be the set of jobs with processing time p. We will show that we can either increase the processing time of all jobs in J_p to $\overline{p}$ or decrease the processing time of all jobs in J_p to $\underline{p}$ without decreasing $\frac{\mathsf{spt}}{\mathsf{opt}}$.

For $j \in J$, let h_j denote the number of jobs processed after j in σ_{opt} plus 1, i.e. $h_j = n - \sigma_{\mathsf{opt}}(j) + 1$. We consider the effect of increasing the processing times of all jobs in J_p by some $\Delta \in [\underline{p} - p, \overline{p} - p]$ and appropriately modifying the arrival times of the resources to match the new starting times (note that Δ may be negative, in which case we decrease the processing times and starting times). This will increase opt by $\Delta \sum_{j \in J_p} h_j$. Indeed, every time we change the processing time of one job j, the completion time of j and of all jobs after j will be increased by Δ.

Notice that the order of the jobs in the SPT schedule does not change. Consider the SPT schedule before the change. Let job $j \in J$ be any job, let j_0 be the the first job that is processed after the last idle time before the starting time of j, and let $i = \sigma^{\mathsf{spt}}(j_0)$ (if there is no idle time before j, let $i = 1$). Let f_j be the number of jobs $j' \in J_p$ with $\sigma_{\mathsf{opt}}(j') < i$; notice that these are exactly the jobs whose modification affects t_i. Thus, the arrival time t_i is changed by Δ for each of those jobs, so the new arrival time is $t_i + \Delta f_j$. This means that the starting time of job j_0 in the changed SPT schedule is at least $S^{\mathsf{spt}}_{j_0} + \Delta f_j$. Now let g_j be the number of jobs $j' \in J_p$ that are processed in the time interval $[t_i, C^{\mathsf{spt}}_j)$ before the change. For each of those jobs, the processing time is changed by Δ and the job is started at or after $t_i + \Delta f_j$, since the SPT order does not change. Thus, the new completion time of j is at least $C^{\mathsf{spt}}_j + \Delta f_j + \Delta g_j$. Consequently, spt will increase by at least $\sum_{j \in J}(f_j + g_j)\Delta$ if $\Delta > 0$, and decrease by at most $\sum_{j \in J}(f_j + g_j)|\Delta|$ if $\Delta < 0$.

If $\frac{\sum_{j \in J}(f_j + g_j)}{\sum_{j \in J_p} h_j} \geq \frac{\mathsf{spt}}{\mathsf{opt}}$, then increasing the processing times in J_p to $\overline{p}$ will not decrease $\frac{\mathsf{spt}}{\mathsf{opt}}$. Otherwise, decreasing the processing times in J_p to $\underline{p}$ will not decrease $\frac{\mathsf{spt}}{\mathsf{opt}}$. Each time we apply this operation, the number of distinct processing times decreases by 1. Finally, we get an instance where the only processing times are $p_{\min} = 0$ and $p_{\max} = 1$. □

Finally, we modify the order of the jobs in the optimal solution. If σ_{opt} and σ_{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 σ_{spt} first starts to process a job of length 1. On one hand, σ_{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 σ_{spt}, because that would mean idle time in σ_{opt}. Thus, if we move all jobs of length 0 and their corresponding resource arrivals in σ_{opt} to time t, then spt does not change but opt decreases. We may thus assume that schedule σ_{opt} processes every job of length 0 at t.

Let k_1 be the number of jobs of length 0 after the transformations. These are processed at time t in σ_{opt}, and these are exactly the jobs processed before time t in σ_{spt}. Thus, there are k_1 arrival times before t, where σ_{opt} processes jobs of length 1. Let k_1+k_2 be the total number of jobs of length 1.

We conclude that σ_{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 σ_{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 (see Fig. 3). The weighted sums of completion times are then given by

$$\text{opt} = \frac{k_1(k_1+1)}{2} + k_1^2 + k_2k_1 + \frac{k_2(k_2+1)}{2}$$

and

$$\begin{aligned}\text{spt} &= \frac{k_1(k_1-1)}{2} + k_2k_1 + \frac{k_1(k_1+1)}{2} \\ &\quad + (k_1+k_2)k_1 + \frac{k_2(k_2+1)}{2}.\end{aligned}$$

We get

$$\begin{aligned}\frac{3}{2}\text{opt} - \text{spt} &= \frac{k_1^2}{4} + \frac{k_2^2}{4} - \frac{k_1k_2}{2} + \frac{3k_1+k_2}{4} \\ &\geq \frac{(k_1-k_2)^2}{4} \geq 0,\end{aligned}$$

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 $\frac{\text{spt}}{\text{opt}}$ converges to $\frac{3}{2}$ as we have $\text{spt} = \frac{9}{2}k_1^2 + \mathcal{O}(k_1)$ and $\text{opt} = 3k_1^2 + \mathcal{O}(k_1)$. This concludes the proof of Theorem 2. □

4 The problem $1|rm = 1, p_j = 0|\sum C_jw_j$

In this section we consider the problem $1|rm = 1, p_j = 0|\sum C_jw_j$, another special case of $1|rm = 1|\sum C_jw_j$. The problem clearly is NP-hard even for $q = 2$ as the knapsack problem can be reduced to it. Indeed, maximizing the weight of the items in the knapsack is equivalent to the task of maximizing the weight of jobs that are scheduled at the first resource arrival time. Recall that Kis (2015) gave a FPTAS for $1|rm = 1|\sum C_jw_j$ when there are two resource arrival times.

First we give a 6-approximation for the problem based on a greedy approach. We also describe a more complicated $(4+\varepsilon)$-approximation that illustrates one of the important ideas of the more general PTAS. Then we provide a PTAS for the case when q, the number of resource arrival times is a constant. This algorithm will be used as a subroutine in the PTAS for the general case. Finally, we prove the main result of the paper which is a PTAS for the case of an arbitrary number of resource arrival times.

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

$$\sum_{j:\pi(j)\leq k} a_j \leq \sum_{i\leq k} b_i \tag{1}$$

for all $1 \leq k \leq q$. As we assume that $\sum_i b_i = \sum_j a_j$ holds, this is equivalent to

$$\sum_{j:\pi(j)\geq k} a_j \geq B_k \tag{2}$$

for all $1 \leq k \leq q$, where $B_k = \sum_{i\geq k} b_i$. Consider the set of jobs that are not processed before a given time point t_k. Then (2) says that if the resource requirements of these jobs add up to at least B_k, then our schedule is feasible. We will mostly use this latter characterization of feasibility, as our algorithms assign the jobs to later time points first. The intuition is that we can bound the approximation ratio by giving sufficiently good upper bounds for every k on the total weight $W_{\geq k}$ of jobs that are processed at time point t_k or later. We present here one such upper bound, that will be used in the first result of the next section, as well as in Sect. 5. For $1 \leq k \leq q$, let

$$m_k = \min\{w(J') : J' \subseteq J,\ \sum_{j\in J'} a_j \geq B_k\},$$

and let M_k be the set of jobs where the minimum is achieved.

Lemma 1 *Let π be a feasible schedule, and let $W_{\geq k} = \sum_{j:\pi(j)\geq k} w_j$. If $W_{\geq k} \leq \alpha \cdot m_k$ for every $1 \leq k \leq q$, then π is an α-approximation.*

Proof Let π^{opt} be the optimal schedule, and let $W^{\text{opt}}_{\geq k} = \sum_{j:\pi^{\text{opt}}(j)\geq k} w_j$. We can bound the objective value of π by

$$\begin{aligned}\sum_{j\in J} w_jC_j &= \textstyle\sum_{k=1}^{q}(t_k - t_{k-1})W_{\geq k} \leq \alpha \sum_{k=1}^{q}(t_k - t_{k-1})m_k \\ &\leq \alpha \textstyle\sum_{k=1}^{q}(t_k - t_{k-1})W^{\text{opt}}_{\geq k} = \alpha \sum_{j\in J} w_jC^{\text{opt}}_j,\end{aligned}$$

which is α times the objective value of π^{opt}. □

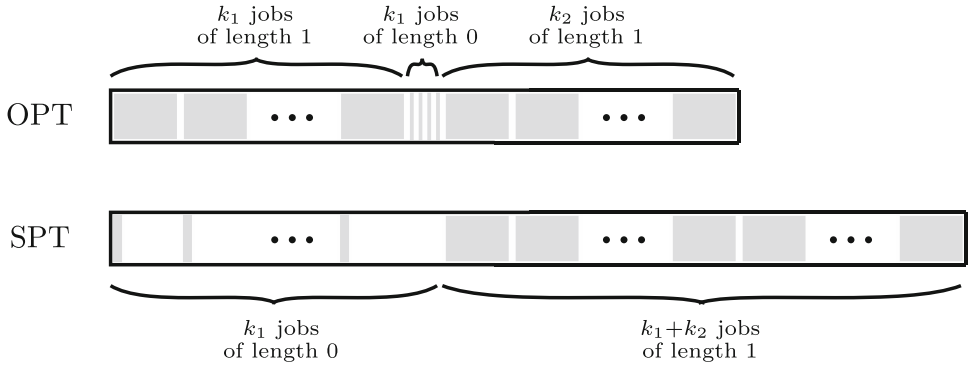

Fig. 3 Schedules σ_{opt} and σ_{spt} after the reductions. The jobs of length 0 are scheduled in σ_{opt} at the first resource arrival time when multiple resources arrive

4.1 A greedy 6-approximation for arbitrary q

The idea of our first algorithm is to have a balance between adding jobs that have small weights and jobs that have high resource requirements. More precisely, we will assign jobs to the time points in reverse order. When we add a job to the set of jobs scheduled after a given time point, we will choose the most inefficient job, i.e. the job minimizing w_j/a_j among all jobs that have weight at most the weight W of all jobs that have already been chosen up to this point. If there is no job with weight at most W, then we simply choose a job with minimal weight. Intuitively, this rule guarantees that the jobs we choose are not too efficient but their total weight is not too large either.

Algorithm 1 Greedy algorithm 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, resource arrival times $t_1 \leq \ldots \leq t_q$ and resource quantities $b_1, \ldots b_q$.
Output: A feasible schedule π.

```
Set A = 0.
Set W = 0.
for i from 0 to q − 1 do
    while A < B_{q−i} do
        if there is an unassigned job j with w_j ≤ W then
            Let j be an unassigned job with w_j ≤ W minimizing w_j/a_j.
        else
            Let j be an unassigned job minimizing w_j.
        W ← W + w_j
        A ← A + a_j
        Set π(j) = q − i.
return π
```

Theorem 3 *For $1|rm = 1, p_j = 0|\sum C_j w_j$, there exists a 6-approximation algorithm with running time $\mathcal{O}(n \log n)$.*

Proof It is clear that the schedule π returned by Algorithm 1 satisfies (2), so it is feasible. We claim that π satisfies the requirements of Lemma 1 with $\alpha = 6$, i.e., $W_{\geq k} \leq 6m_k$ for every k. Let M_k denote the set of jobs where the minimum m_k is achieved.

To bound $W_{\geq k}$ for a given k, we consider the algorithm up until the last step where $q - i \geq k$, and divide it into 3 phases (some of these may be empty):

- Phase 1: until the first iteration where W becomes at least m_k
- Phase 2: until the last iteration where $A < B_k$
- Phase 3: the last step, where A becomes at least B_k.

At the end of phase 1, we have $W \leq 2m_k$, because at the beginning of the last step of phase 1, we either add a job of weight at most W (which is less than m_k at that point), or we add a job of minimum weight, which is again at most m_k because some job in M_k is still unassigned.

In phase 2, the total weight added is at most m_k. Indeed, throughout phase 2, some job in M_k is still unassigned, but $W \geq m_k$, so we always pick an unassigned job j with $w_j \leq W$ minimizing w_j/a_j. Thus, the job selected is at least as inefficient as any unassigned job in M_k, so the total weight of the selected jobs cannot be larger than m_k.

At the beginning of phase 3, we have $A < B_k$ and $m_k \leq W \leq 3m_k$. At this point, there is still an unassigned job in M_k, so the selected job has weight at most $W \leq 3m_k$. Thus, the total weight is at most $6m_k$.

The running time bound follows by ordering the jobs according to their weight and by using AVL trees for picking j in the *while* loop. □

The following example shows that the bound is tight. We have 5 jobs with weights $w_1 = w_2 = 1 - 2\varepsilon$, $w_3 = 1 - \varepsilon$, $w_4 = 1$ and $w_5 = 3$. The resource requirements are $a_1 = a_2 = \varepsilon/5$, $a_3 = 1 - \varepsilon/2$, $a_4 = 1$ and $a_5 = 4$. The resource arrival times are $t_1 = 0$ and $t_2 = 1$, with resource quantities $b_1 = 5 - \varepsilon/10$ and $b_2 = 1$. Here the optimum solution is to schedule the job with weight 1 to time point t_2 and all the remaining jobs to time point t_1. However, our algorithm will schedule the job with weight 1 to time point t_1 and all the remaining jobs to t_2.

4.2 A $(4+\varepsilon)$-approximation for arbitrary q

Now we give a slightly better approximation for the problem. The algorithm is a bit more complicated than the one presented in Sect. 4.1, but the proof illustrates one of the important ideas of the general PTAS.

The idea of the algorithm is as follows. We may assume without loss of generality that resource arrival times are integer, because multiplying all arrival times by a large integer does not change the problem. First we shift all resource arrival times to powers of 2. For each arrival time t_i in the shifted instance, we apply the FPTAS by Kis (2015) to the instance which has only two resource arrival times t_1 and t_i, and the resource quantity for t_i is B_i. Denote the set of jobs assigned to t_i this way by L_i. Then, we schedule each job j at the latest time point t_i where $j \in L_i$, i.e. $\pi(j) = \max\{i : j \in L_i\}$.

More formally, let $\mathcal{I}$ be an instance of $1|rm = 1, p_j = 0|\sum_j C_j w_j$. We assume $t_1 = 0$ and $t_2 = 1$ (the latter assumption is without loss of generality because we can add an arrival time with 0 resource arrival). 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

$$t'_i = \begin{cases} 0 & \text{if } i = 1, \\ 2^{i-2} & \text{for } i = 2, \dots, \lceil \log_2(t_q) \rceil + 2, \end{cases}$$

and

$$b'_i = \begin{cases} b_i & \text{if } i = 1, 2, \\ \sum[b_\ell : t_\ell \in (2^{i-3}, 2^{i-2}]\,] & \text{for } i = 3, \dots, \lceil \log_2(t_q) \rceil + 2. \end{cases}$$

Claim 5 *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 $2W$. Furthermore, any feasible schedule for $\mathcal{I}'$ is also feasible for $\mathcal{I}$.*

Proof Let us define $t_i^* = \min\{t'_\ell : t_i \le t'_\ell\}$ for $i = 1, \dots, q$. Let π be the solution for $\mathcal{I}$. Then assigning all jobs that are assigned to time point t_i to t_i^* gives us a feasible solution to $\mathcal{I}'$. By this change, the completion of any job is at most doubled (recall that each t_i is assumed to be integer).

Since the available amount of resources at each time in $\mathcal{I}'$ is at most as much as in $\mathcal{I}$, a feasible schedule for $\mathcal{I}'$ is also a feasible schedule for $\mathcal{I}$. □

Claim 6 *There exists a polynomial time $(2+\varepsilon)$-approximation algorithm for constant ε for all instances $\mathcal{I}$ where the resource arrival times are integer powers of 2.*

Proof We use the procedure that we described above, i.e., for each $i > 1$ we solve the instance with B_i resource arriving at t_i and the rest at t_1, using the FPTAS provided by Kis (2015). As defined above, L_i is the set of jobs assigned to t_i by the FPTAS, and $\pi(j) = \max\{i : j \in L_i\}$.

Let $\alpha = 1 + \varepsilon$. Let π^{opt} be an optimum solution and let J_k^{opt} be the set of jobs j with $\pi^{\mathsf{opt}}(j) = k$. We have

$$w(L_i) \le \alpha \sum_{k=i}^{q} w(J_k^{\mathsf{opt}})$$

for $i = 1, \dots, q$. Then we get

$$\begin{aligned} 2\alpha \sum_{j \in J} w_j C_j^{\pi^{\mathsf{opt}}} &= \sum_{i=2}^{q} (2\alpha) \cdot 2^{i-2} w(J_i^{\mathsf{opt}}) \\ &= \sum_{i=2}^{q} \alpha \cdot 2^{i-2} w(J_i^{\mathsf{opt}}) \left(1 + \sum_{j=1}^{\infty} 2^{-j}\right) \\ &\ge \sum_{i=2}^{q} \alpha 2^{i-2} \sum_{k=i}^{q} w(J_k^{\mathsf{opt}}) \\ &\ge \sum_{i=2}^{q} 2^{i-2} w(L_i), \end{aligned}$$

thus the approximation ratio follows. □

The two claims show that this approach leads to a $(4+\varepsilon)$-approximation with running time polynomial in $1/\varepsilon$ and the input length.

4.3 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.

Theorem 4 *Consider the number of arrival times q to be constant. For any fixed positive integer k, there is a $(1 + \frac{q}{k})$-approximation algorithm for $1|rm = 1, p_j = 0|\sum C_j w_j$ with running time $\mathcal{O}(n^{qk+1})$.*

Proof We claim that Algorithm 2 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 H_i^{opt} be the set of the k heaviest jobs in J_i^{opt} if $|J_i^{\mathsf{opt}}| \ge k$, otherwise let $H_i^{\mathsf{opt}} = J_i^{\mathsf{opt}}$. Let $J_i = \{j \in J : \pi(j) = i\}$ denote the set of jobs

Algorithm 2 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 $H_1 \cup \cdots \cup H_q \subseteq J$ with $|H_i| \leq k$ for $i > 1$ **do**
2: Set $A = 0$.
3: Set $W = 0$.
4: **for** i from 0 to $q-2$ **do**
5: **for** $j \in H_{q-i}$ **do**
6: $\pi(j) = q - i$
7: $A \leftarrow A + a_j$
8: **if** $|H_{q-i}| = k$ **then**
9: $W \leftarrow \max\{W, \min\{w_j : j \in H_{q-i}\}\}$
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

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 $H_1^{\mathsf{opt}} \cup \cdots \cup H_q^{\mathsf{opt}}$ 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. To show feasibility of π, observe that any job $j \notin H_1^{\mathsf{opt}} \cup \cdots \cup H_q^{\mathsf{opt}}$ for which $\pi^{\mathsf{opt}}(j) \geq q - \ell$ always satisfies $w_j \leq W_{q-\ell}$, so we can pick jobs in line 11 until $A \geq B_{q-\ell}$.

Now we prove the approximation factor. 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 H_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. □

4.4 PTAS for arbitrary q

We turn to the proof of the main result of the paper. As in Sect. 4.2, we shift resource arrival times; here we use 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

$$t'_i = \begin{cases} 0 & \text{if } i = 1, \\ (1+\varepsilon)^{i-2} & \text{for } i = 2, \ldots, \lceil \log_{1+\varepsilon}(t_q) \rceil + 2, \end{cases}$$

and

$$b'_i = \begin{cases} b_i & \text{if } i = 1, 2, \\ \sum [b_\ell : t_\ell \in ((1+\varepsilon)^{i-3}, (1+\varepsilon)^{i-2}]] & \text{for } i = 3, \ldots, \lceil \log_{1+\varepsilon}(t_q) \rceil + 2. \end{cases}$$

The proof of the following claim is the same as that of Claim 5.

Claim 7 *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 $\mathcal{I}'$ also is a feasible schedule for $\mathcal{I}$.*

Due to the claim, we may assume that the positive arrival times are powers of $1 + \varepsilon$. For convenience of notation, in this section we will assume 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.

Theorem 5 *There exists a PTAS for* $1|rm = 1, p_j = 0|\sum C_j w_j$.

Proof 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, and are indexed in decreasing order.

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 $r+1$ 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 $\ell \in \{1, \dots, r/2\}$. See Algorithm 3.

Algorithm 3 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 2 on $\mathcal{I}'$ to get schedule π'.
  Let $A = B = 0$
  for i from 0 to $\ell - 1$ **do**
    For every j for which $\pi'(j) = i$, fix $\pi_\ell(j) = i$
    $A \leftarrow A + \sum\{a_j : \pi'(j) = i\}$
    $B \leftarrow B + b_i$
  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 2 on $\mathcal{I}'$ to get schedule π'.
    for i from s to $s + r/2 - 1$ **do**
      For every j for which $\pi'(j) = i$, fix $\pi_\ell(j) = i$
      $A \leftarrow A + \sum\{a_j : \pi'(j) = i\}$
      $B \leftarrow B + b_i$
  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

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

$$\sum_{i=0}^{\ell-1} (1+\varepsilon)^{-i} W_i + \sum_{i=\ell}^{\ell+r/2-1} (1+\varepsilon)^{-i} W_i' \leq (1+\varepsilon) \sum_{i=0}^{\ell+r/2-1} (1+\varepsilon)^{-i} W_i^*,$$

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 $i = s, s+1, \dots, s+r/2-1$, total weight W_i^* at t_i for $i = s + r/2 + 1, \dots, s + r - 1$, 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

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

The sum of these inequalities gives

$$\sum_{i=0}^{q-2} (1+\varepsilon)^{-i} W_i \leq \varepsilon \sum_{i=\ell}^{q-2} (1+\varepsilon)^{-i} W_i' + (1+\varepsilon) \sum_{i=0}^{q-2} \times (1+\varepsilon)^{-i} W_i^* x + (1+\varepsilon) \sum_{i=0}^{q-2}$$

$$\times \left(\sum_{j:jr/2+\ell>i} (1-\varepsilon)^{-(jr/2+\ell)} \right) W_i^*. \tag{3}$$

To bound the first term on the right hand side of (3), 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,

$$\begin{aligned}
&\sum_{i=(j+1)r/2+\ell}^{(j+2)r/2+\ell-1} (1+\varepsilon)^{-i} W_i' \\
&\quad \le (1+\varepsilon) \left(\sum_{i=jr/2+\ell}^{(j+2)r/2+\ell-1} (1+\varepsilon)^{-i} W_i^* \right. \\
&\qquad \left. +(1+\varepsilon)^{-jr/2-\ell} \sum_{i=0}^{jr/2+\ell-1} W_i^* \right),
\end{aligned}$$

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

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

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

The last term of the right side of (3) 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

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

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

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

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. □

5 Undetermined resource arrival times

In this section, we consider a variant of the problem where it is known that the arriving resource quantities will be $b_1, \ldots, b_q$ in this order, but the resource arrival times t_j are unknown. A solution (schedule) consists of an ordering of the jobs, which determines the completion times for any fixed sequence of resource arrival times; this is then compared to the optimum for those arrival times. The problem is denoted by $1|rm = 1, p_j = 0, t_i \text{ unknown}| \sum C_j w_j$.

Consider an instance of the problem. Recall that $B_k = \sum_{i \ge k} b_i$, M_k denotes the minimum weight job set consuming at least B_k resources, and $m_k = w(M_k)$. Let π be a feasible schedule, and let $W_{\ge k} = \sum_{j:\pi(j) \ge k} w_j$. Lemma 1 stated that if $W_{\ge k} \le \alpha \cdot m_k$ for every $1 \le k \le q$, then π is an α-approximation for given arrival times. However, the condition does not depend on the arrival times, so a schedule π satisfying the condition is actually an α-approximation for this instance of $1|rm = 1, p_j = 0, t_i \text{ unknown}| \sum C_j w_j$.

Conversely, the smallest α for which a feasible schedule satisfying the condition of Lemma 1 exists is the best approximation ratio possible; this can be seen by choosing the k for which $W_{\geq k}$ is the worst approximation, and setting the arrival time t_i to 0 if $i < k$ and to 1 if $i \geq k$.

The proof of Theorem 3 implies that Algorithm 1 is a 6-approximation algorithm for $1|rm = 1, p_j = 0, t_i$ unknown$|\sum C_j w_j$. In the following, we show that for any instance, there is a schedule which gives a 4-approximation, while for $\alpha < 4$ there are instances where no such schedule exists. We also show that a $(4+\varepsilon)$-approximation can be found efficiently for any ε; it remains open whether 4-approximation is possible in polynomial time.

Theorem 6 *For $1|rm = 1, p_j = 0, t_i$ unknown$|\sum C_j w_j$, there exists a $(4+\varepsilon)$-approximation with running time polynomial in $1/\varepsilon$ and the input length. Moreover, there is no $(4-\varepsilon)$-approximation algorithm for the problem for any $\varepsilon > 0$.*

Proof Our approximation algorithm is based on the following claim.

Claim 8 *For any instance of $1|rm = 1, p_j = 0, t_i$ unknown$|\sum C_j w_j$, there exists a schedule π such that $W_{\geq k} \leq 4m_k$ for every $1 \leq k \leq q$.*

Proof Define $f(i) = \min\{k : w(M_k) \leq 2w(M_i)\}$ for $i = 2, \ldots, q$ and let us consider the following procedure (Algorithm 4).

Algorithm 4 Subroutine for $(4+\varepsilon)$-approximation to $1|rm = 1, p_j = 0, t_i$ unkown$|\sum C_j w_j$.

Input: Jobs J with $|J| = n$, resource requirements a_j, weights w_j, resource quantities $b_1, \ldots b_q$.
Output: A feasible schedule π.
1: Set $i = q$.
2: **while** $i \geq 1$ **do**
3: Set $L_{i+1} = \{j : \pi(j) > i\}$.
4: Set $\pi(j) = i$ for $j \in M_{f(i)} \setminus M_{i+1}$.
5: $i \leftarrow f(i) - 1$
6: **return** π

It is not difficult to see that π is a feasible schedule. We prove by induction that $W_{\geq i} = w(L_i) \leq 4w(M_i)$ holds for $i = 1, \ldots, q$. As $L_q = M_{f(q)}$, the inequality $w(L_q) \leq 4w(M_q)$ clearly holds. Assume now that $i \leq q-1$. If no jobs are assigned to t_i, then $w(L_i) = w(L_{i+1}) \leq 4w(M_{i+1}) \leq 4w(M_i)$. Otherwise $i = f(i') - 1$, where i' is the index considered in the previous iteration of the while loop in Step 2. Observe that no jobs are assigned to time points between the ith and the i'th ones. By induction, we get

$$w(L_i) = w(M_{f(i)} \cup L_{i'}) \leq w(M_{f(i)}) + w(L_{i'}) \leq 2w(M_i) + 4w(M_{i'}) \leq 4w(M_i).$$

Here the second inequality holds by induction and by the definition of f, while the last inequality follows from the fact that $i < f(i')$ which implies $w(M_i) > 2w(M_{i'})$. □

Now we show how Algorithm 4 provides a $(4+\varepsilon)$-approximation for the problem $1|rm = 1, p_j = 0, t_i$ unkown$|\sum C_j w_j$ which has running time polynomial in the input size and $\frac{1}{\varepsilon}$. Using either an FPTAS for the knapsack problem or the FPTAS of Kis (2015), we determine an approximation of the sets M_i. Then we apply Algorithm 4 with these approximations in place of the sets M_i to schedule the jobs. This concludes the proof of the first part of the theorem.

The following set of instances shows that α is at least 4. We are given $(n-1)m$ jobs denoted by $1, 2, \ldots, (n-1)m$ with weights $w_i = n - \frac{i}{m}$ and $a_i = 2^{-i}$. Furthermore, we have $(n-1)m$ resource arrival times that are unknown. The resource quantities are given by $b_q = 2^{-q}$ and $b_i = 2^{-i} - B_{i+1} = 2^{-i-1}$ for $1 < i < q$; b_1 equals the remaining resource requirement.

In order to fulfill the resource requirements, the set of jobs scheduled at or after time t_i has to contain at least one of the jobs $j \leq i$. Observe that the optimal solution of finding a minimum weight job set M_i consuming at least B_i resources consists of the single job i.

Let j_1 be a job processed at time t_q. We may assume that $w_{j_1} \leq 4w_q = 4$, because otherwise this schedule is not a 4-approximation if $t_q = 1$ and all other arrival times are set to 0.

We create a sequence starting with j_1. Since the jobs with indices greater or equal to j_1 have total resource requirement less than B_{j_1-1}, we have to schedule at least one job $j_2 < j_1$ at or after t_{j_1-1}. This argument can be iterated to find a job $j_3 < j_2$ which is scheduled at or after t_{j_2-1}, and so on, until $j_N = 1$ for some N.

The following claim shows that for n, m large enough, there must be an index i such that the jobs in this sequence that are scheduled at or after t_{j_i-1} have large total weight compared to $w_{j_i-1} = w(J_{j_i-1})$.

Claim 9 *For any $\beta < 4$, there exist n and m such that for any feasible schedule, there is some i with*

$$\beta w_{j_i-1} < \sum_{k=1}^{i+1} w_{j_k}$$

for the sequence $j_1, j_2, \ldots$ constructed as above.

Proof Suppose to the contrary that there is no triple n, m, i satisfying the requirements of the claim. Then for all n, m, there is a feasible schedule such that $\beta w_{j_i-1} \geq \sum_{k=1}^{i+1} w_{j_k}$ for all i.

By setting m large enough, $w_{j_i-1} - w_{j_i} = 1/m$ is very small; we will see that $m > 12/(4-\beta)$ is enough. By increasing n, the length N of our sequence increases as well. Indeed,

by the indirect assumption, we have

$$w_{j_{i+1}} \leq \sum_{k=1}^{i+1} w_{j_k} \leq \beta w_{j_i-1} < 4w_{j_i-1} = 4w_{j_i} + 4/m.$$

Since we also have $w_{j_1} \leq 4$, $w_j \geq 1$ for all jobs j, and $w_1 = n$, this implies that $N \geq \log_5 n$. Let us define $z_i = w_{j_i}$ and $z'_i = \sum_{k=1}^{i-1} z_k$. By the indirect assumption, $\beta(z_i + \frac{1}{m}) \geq z_{i+1} + z_i + z'_i$, thus we have

$$(\beta - 1)z_i - z'_i + 4/m > z_{i+1}. \tag{4}$$

We claim that

$$\frac{z'_{i+1}}{z_{i+1}} > \frac{3}{\beta - 1}\frac{z'_i}{z_i} \tag{5}$$

for every i, or equivalently,

$$(\beta - 1)(z_i^2 + z_i z'_i) - 3z'_i z_{i+1} > 0.$$

By (4), the left hand side is at least

$$\begin{aligned}&(\beta - 1)(z_i^2 + z_i z'_i) - 3z'_i((\beta - 1)z_i - z'_i + 4/m)\\ &\quad = (\beta - 1)z_i^2 - 2(\beta - 1)z_i z'_i + 3z'_i(z'_i - 4/m).\end{aligned}$$

This is positive if $3(z'_i - 4/m) > (\beta - 1)z'_i$, which holds if m is large enough (e.g. $m > 12/(4 - \beta)$), since $z'_i \geq 1$ and $\beta - 1 < 3$.

Since $3/(\beta - 1) > 1$, it follows from (5) that $\frac{z'_N}{z_N} > 3$ for large enough N, and thus we get

$$\beta w_{j_{N-1}-1} \leq \beta z_N < 4z_N \leq z_N + z'_N,$$

contradicting the indirect assumption. □

By Claim 9, there exists a time point t_i with $\sum_{j:\pi(j)\geq i} w_j > \beta w(J_i)$. By setting $t_{i'} = 0$ (or very close to 0) for $i' < i$ and $t_{i'} = 1$ (or very close to 1) for $i' \geq i$, the schedule can only be a $(\beta - \varepsilon)$-approximation if we do not know the resource arrival times in advance. □

Funding Open access funding provided by Eötvös Loránd University.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Carlier, J. (1984). *Problèmes d'ordonnancement à contraintes de ressources: algorithmes et complexité*. Institut de programmation: Université Paris VI-Pierre et Marie Curie.

Carlier, J., & Kan, A. R. (1982). Scheduling subject to nonrenewable-resource constraints. *Operations Research Letters, 1*(2), 52–55.

Gafarov, E. R., Lazarev, A. A., & Werner, F. (2011). Single machine scheduling problems with financial resource constraints: Some complexity results and properties. *Mathematical Social Sciences, 62*(1), 7–13.

Garey, M. R., & Johnson, D. S. (1979). Computers and intractability. *A guide to the theory of NP-completeness*.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics, 5*, 287–326.

Grigoriev, A., Holthuijsen, M., & Van De Klundert, J. (2005). Basic scheduling problems with raw material constraints. *Naval Research Logistics (NRL), 52*(6), 527–535.

Györgyi, P. (2017). A PTAS for a resource scheduling problem with arbitrary number of parallel machines. *Operations Research Letters, 45*(6), 604–609.

Györgyi, P., & Kis, T. (2014). Approximation schemes for single machine scheduling with non-renewable resource constraints. *Journal of Scheduling, 17*(2), 135–144.

Györgyi, P., & Kis, T. (2015a). Approximability of scheduling problems with resource consuming jobs. *Annals of Operations Research, 235*(1), 319–336.

Györgyi, P., & Kis, T. (2015b). Reductions between scheduling problems with non-renewable resources and knapsack problems. *Theoretical Computer Science, 565*, 63–76.

Györgyi, P., & Kis, T. (2017). Approximation schemes for parallel machine scheduling with non-renewable resources. *European Journal of Operational Research, 258*(1), 113–123.

Györgyi, P., & Kis, T. (2018). Minimizing the maximum lateness on a single machine with raw material constraints by branch-and-cut. *Computers & Industrial Engineering, 115*, 220–225.

Györgyi, P., & Kis, T. (2019). Minimizing total weighted completion time on a single machine subject to non-renewable resource constraints. *Journal of Scheduling, 22*(6), 623–634.

Györgyi, P., & Kis, T. (2020). 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.

Kis, T. (2015). Approximability of total weighted completion time with resource consuming jobs. *Operations Research Letters, 43*(6), 595–598.

Slowiński, R. (1984). Preemptive scheduling of independent jobs on parallel machines subject to financial constraints. *European Journal of Operational Research, 15*(3), 366–373.

Toker, A., Kondakci, S., & Erkip, N. (1991). Scheduling under a non-renewable resource constraint. *Journal of the Operational Research Society, 42*(9), 811–814.

Xie, J. (1997). Polynomial algorithms for single machine scheduling problems with financial constraints. *Operations Research Letters, 21*(1), 39-42.