Noname manuscript No.
(will be inserted by the editor)

Approximability of scheduling problems with resource consuming jobs

Péter Györgyi · Tamás Kis

Received: date / Accepted: date

Abstract The paper presents new approximability results for single machine scheduling problems with jobs requiring some non-renewable resources (like raw materials, energy, or money) beside the machine. Each resource has an initial stock and additional supplies over time. A feasible schedule specifies a starting time for each job such that no two jobs overlap in time, and when a job is started, enough resources are available to cover its requirements. The goal is to find a feasible schedule of minimum makespan. This problem is strongly NP-hard.

Recently, the authors of this paper have proposed a PTAS for the special case with a single non-renewable resource and with a constant number of supply dates, as well as an FPTAS for the special case with two supply dates and one resource only. In this paper we prove APX-hardness of the problem when the number of resources is part of the input, and new polynomial time approximation schemes are devised for some variants, including (1) job release dates, and more than one, but constant number of resources and resource supply dates, and (2) only one resource, arbitrary number of supply dates and job release dates, but with resource requirements proportional to job processing times.

Keywords Single machine scheduling, non-renewable resources, approximation schemes, vertex cover problem

1 Introduction

In this paper we study scheduling problems with resource consuming jobs. In these problems there are non-renewable resources (like raw materials, energy, or money) consumed by the jobs. Each non-renewable resource has an initial stock, which is replenished at a-priori known moments of time and in known quantities. We will consider only single-machine problems, i.e., all the jobs have to be sequenced on the same machine. The sole optimization objective will be the schedule length

P. Györgyi · T. Kis
Institute for Computer Science and Control, H1111 Budapest, Kende str. 13–17, Hungary
Tel.: +36 1 2796156; Fax: +36 1 4667503
E-mail: gyorgyi.peter@sztaki.mta.hu, kis.tamas@sztaki.mta.hu

P. Györgyi
Eötvos Loránd University, Pázmány Péter Sétány 1/C, Budapest, Hungary, H1117

(makespan). Our analysis relies on connections with variants of the knapsack problem, and with the vertex cover problem in graphs.

More formally, there is a single machine, a finite set of jobs $\mathcal{J}$, and a finite set of non-renewable resources $\mathcal{R}$ consumed by the jobs. The machine can perform only one job at a time, and preemption is not allowed. Each job J_j has a processing time $p_j \in \mathbb{Z}_+$, a release date r_j, and resource requirements $a_{ij} \in \mathbb{Z}_+$ from the resources $i \in \mathcal{R}$. The resources are supplied in q different moments in time, $0 = u_1 < u_2 < \ldots < u_q$; the vector $\tilde{b}_\ell \in \mathbb{Z}_+^{|\mathcal{R}|}$ represents the quantities supplied at u_ℓ. A *schedule* σ specifies the starting time S_j of each job and it is *feasible* if (i) the jobs do not overlap in time, (ii) $S_j \geq r_j$ for all $j \in \mathcal{J}$, and if (iii) at any time point t the total material supply from every resource is at least the total request of those jobs starting not later than t, i.e., $\sum_{(\ell \,:\, u_\ell \leq t)} \tilde{b}_{\ell i} \geq \sum_{(j \,:\, S_j \leq t)} a_{ij},\ \forall i \in \mathcal{R}$. The objective is to minimize the makespan, i.e., the completion time of the job finished last.

Assumption 1 $\sum_{\ell=1}^{q} \tilde{b}_{\ell i} = \sum_{j \in \mathcal{J}} a_{ij},\ \forall i \in \mathcal{R}$, *holds without loss of generality.*

Assumption 1 implies that there must exist a feasible solution (if every job starts not before u_q, the last supply date) and at least one job must start not before u_q (thus the optimal makespan $C^*_{\max}$ is greater than u_q).

1.1 Previous work

Scheduling problems with resource consuming jobs were introduced by Carlier (1984), and Carlier and Rinnooy Kan (1982). Further results can be found in e.g., Slowinski (1984), Toker et al. (1991), Neumann and Schwindt (2002), Laborie (2003), Grigoriev et al. (2005), Briskorn et al. (2010, 2013), Gafarov et al. (2011), Györgyi and Kis (2014, 2015), Morsy and Pesch (2015). In particular, in Grigoriev et al. (2005) and Gafarov et al. (2011) the complexity of several variants was studied and some constant ratio approximation algorithms were developed in Grigoriev et al. (2005). Briskorn et al. (2010), Briskorn et al. (2013) and Morsy and Pesch (2015) examined scheduling problems where there is an initial inventory, and no more supplies, but some of the jobs produce resources, while other jobs consume the resources. In Briskorn et al. (2010) and Briskorn et al. (2013) problems with the objective of minimizing the inventory levels were studied. Morsy and Pesch (2015) designed approximation algorithms to minimize the total weighted completion time. In Györgyi and Kis (2014) a PTAS for scheduling resource consuming jobs with a single non-renewable resource and a constant number of supply dates was developed, and also an FPTAS was devised for the special case with $q = 2$ supply dates and one non-renewable resource only. In Györgyi and Kis (2015) it was shown, among other results, that there is no FPTAS for the problem of scheduling jobs on a single machine with two non-renewable resources and $q = 2$ supply dates, unless $P = NP$, which is in strong contrast with the existence of an FPTAS for the special case with one non-renewable resource only (Györgyi and Kis, 2014). In Györgyi and Kis (2015) and Györgyi and Kis (2014), variants of the knapsack problem are solved as a subproblem using combinatorial techniques like enumeration of feasible packings. However, an important and very fruitful algorithmic technique for solving packing type problems is linear programming based rounding, see e.g., Fleischer et al. (2011), that will be used in this paper as well.

While there are some algorithmic results for the more general resource constrained project scheduling problem (RCPSP) with non-renewable resources, see e.g., Neumann and Schwindt (2002), Laborie (2003), but, to our best knowledge, the hardness of approximation has only been studied for RCPSP without any non-renewable resources, see Gafarov et al. (2014).

1.2 Results of the paper

Our positive and negative results are presented in the following two subsections.

1.2.1 Non-approximability results

If the number of non-renewable resources is constant and the number of supply dates is 2, then the problem $1|rm = const., q = 2|C_{\max}$ admits a PTAS (Györgyi and Kis, 2015). In contrast, if the number of resources is part of the input, we can prove the following result.

Theorem 1 *Unless $P = NP$, there is some constant $\varepsilon > 0$ such that it is NP-hard to approximate the problem $1|rm, q = 2|C_{\max}$ better than $1 + \varepsilon$ if the number of resources is part of the input.*

Since the problem $1|rm|C_{\max}$ admits a 2-approximation algorithm (Grigoriev et al., 2005), and $1|rm, q = 2|C_{\max}$ is just a special case, we can deduce the following:

Corollary 1 *$1|rm, q = 2|C_{\max}$ is APX-complete.*

It is also known that if the number of resources is constant and at least 2, then there is no FPTAS for $1|rm = const., q = 2|C_{\max}$.

1.2.2 Approximation schemes

Our new approximation schemes can solve more general problems than $1|rm = 1, q = const|C_{\max}$, for which a PTAS has been developed in Györgyi and Kis (2014). On the one hand, we allow more than one, but a constant number of resources, and on the other hand, we consider job release dates as well.

Theorem 2 *There is a PTAS for the problem $1|rm = const., q = const., \#\{r_j : r_j < u_q\} = const.|C_{\max}$.*

The condition $\#\{r_j : r_j < u_q\} = const.$ reads that the number of distinct job release dates before u_q is bounded by a constant. The new PTAS inherits some of the components from the earlier result, like scheduling small and big jobs separately, but in this paper we use linear programming based rounding to schedule the small jobs, and in the analysis we prove only that the rounding is just good enough to get the desired approximation for the original scheduling problem, instead of the stronger result proved in Györgyi and Kis (2014) showing that the scheduling of the small jobs is a good approximation for a subproblem similar to the multiple knapsack problem. The following result dispenses with the condition on the number of distinct job release dates.

Theorem 3 *There is a PTAS for the problem* $1|rm = const., q = const.|C_{\max}$.

The proof of this results uses a rounding argument, and relies on the PTAS developed for proving Theorem 2. Finally, we can get rid of the constant bound on the number of resource supplies at the expense of considering one resource only and restricting the resource requirements to be proportional to the job processing times, i.e., there exists a positive constant λ such that $a_j = \lambda p_j$ for all $j \in \mathcal{J}$. The constant λ of course depends on the problem instance. This assumption may be quite reasonable in some practical applications. Since we can get an equivalent problem by dividing all the supplies, and all the resource requirements of a problem instance by the (instance specific) constant λ, from now on we consider the case $a_j = p_j$ only. Notice that in the above transformation, the $\tilde{b}_\ell$ may become fractional after dividing by λ. However, this does not create any difficulty for the approximation algorithms proposed below.

Theorem 4 *There is a PTAS for the problem* $1|rm = 1, a_j = p_j|C_{\max}$.

One can generalize this result by enabling job specific release dates as well.

Theorem 5 *There is a PTAS for the problem* $1|rm = 1, p_j = a_j, r_j|C_{\max}$.

In Table 1 we summarize the results of the paper, and for the sake of completeness, we also mention previous results for this class of problems.

#Supplies q	#Resources rm	Release dates r_j	PTAS	FPTAS
2	1	no	yes (Györgyi and Kis, 2014)	yes (Györgyi and Kis, 2014, 2015)
2	1	yes	yes (Sect. 5)	?
2	*const.* ≥ 2	no	yes (Györgyi and Kis, 2014)	no[a] (Györgyi and Kis, 2015)
2	*const.* ≥ 2	yes	yes (Sect. 5)	no[a] (Györgyi and Kis, 2015)
2	arbitrary	yes/no	no[a] (Sect. 3)	no[a] (Sect. 3)
const. ≥ 3	1	yes/no	yes (Sect. 5)	?
const. ≥ 3	*const.* ≥ 2	yes/no	yes (Sect. 5)	no[a] (Györgyi and Kis, 2015)
arbitrary	1	no	yes[b] (Sect. 6)	no[a] (Grigoriev et al., 2005)
arbitrary	1	yes	yes[b] (Sect. 7)	no[a] (Grigoriev et al., 2005)

[a] if $\mathcal{P} \neq \mathcal{NP}$

[b] under the condition $a_j = p_j$

Table 1 Known approximability results for scheduling problems with resource consuming jobs. In the column of release dates "yes / no" means that the result is valid in both cases. The question mark "?" indicates that we are not aware of any definitive answer.

1.3 Structure of the paper

In Section 2 we provide a problem formulation in terms of a mathematical program which will be used throughout the paper. In Sections 3, 4, 5, 6, and 7 we prove Theorems 1, 2, 3, 4 and 5, respectively. Some final remarks and open questions are collected in Section 8.

1.4 Terminology and definitions

An *optimization problem* Π consists of a set of instances, where each instance has a set of *feasible solutions*, and each solution has a cost. In a *minimization problem* a feasible solution of minimum cost is sought, while in a maximization problem one of maximum cost. The value of the best (or optimal) solution of instance x of Π is denoted by $\mathrm{opt}(x)$. An *ε-approximation algorithm* for an optimization problem Π delivers in polynomial time for each instance of Π a solution whose objective function value is at most $(1+\varepsilon)$ times the optimum value in case of minimization problems, and at least $(1-\varepsilon)$ times the optimum in case of maximization problems. For an optimization problem Π, a family of approximation algorithms $\{A_\varepsilon\}_{\varepsilon>0}$, where each A_ε is an ε-approximation algorithm for Π is called a *Polynomial Time Approximation Scheme (PTAS) for Π*. If, in addition, each A_ε in the family is of polynomial time in $1/\varepsilon$ as well, then $\{A_\varepsilon\}_{\varepsilon>0}$, is called a *Fully Polynomial Time Approximation Scheme (FPTAS) for Π*. The *class PTAS / class FPTAS* consists of those optimization problems which admit a polynomial time approximation scheme / fully polynomial time approximation scheme. The above definitions are mainly from the book of Garey and Johnson (1979).

The *class APX* consists of those optimization problems that can be approximated within some constant factor in polynomial time in the size of the input. Clearly, the class PTAS is a subset of the class APX. The *class APX-complete* comprises those problems from APX which do not belong to PTAS, unless P=NP. Let Π_1 and Π_2 be two optimization problems. We say that Π_1 *L-reduces* to Π_2 if there exist two polynomial time algorithms f and g, and two constants $\alpha, \beta > 0$, such that for every instance x of Π_1:

i) $\mathrm{opt}_{\Pi_2}(f(x)) \leq \alpha \cdot \mathrm{opt}_{\Pi_1}(x)$,
ii) for any solution of $f(x)$ with cost c_2, g provides a solution of x with cost c_1 such that $|c_1 - \mathrm{opt}_{\Pi_1}(x)| \leq \beta \cdot |c_2 - \mathrm{opt}_{\Pi_2}(f(x))|$.

The two most important properties of L-reductions are that they compose, and if Π_1 L-reduces to Π_2, and there is an ε-approximation algorithm for Π_2, then, through the reduction, we get an $\alpha\beta\varepsilon$-approximation algorithm for Π_1. See the original paper by Papadimitriou and Yannakakis (1991) for more details.

2 Problem formulation

We can model our scheduling problem by means of a mathematical program. To this end, firstly we construct a set of time points $\mathcal{T}$ consisting of all the *distinct values* from the set of time moments u_ℓ, $\ell = 1, \ldots, q$, (when some non-renewable resource is supplied), and the set of release dates of the jobs r_j, $j \in \mathcal{J}$. Suppose $\mathcal{T}$ has τ elements, denoted by v_1 through v_τ, with $v_1 = 0$. We define the values $b_{\ell i} := \sum_{k \,:\, u_k \leq v_\ell} \tilde{b}_{ki}$ for $i \in \mathcal{R}$, that is, $b_{\ell i}$ equals the total amount supplied from resource i up to time point v_ℓ.

We introduce $\tau \cdot |\mathcal{J}|$ binary decision variables $x_{j\ell}$, $(j \in \mathcal{J}, \ell = 1, \ldots, \tau)$ such that $x_{j\ell} = 1$ if and only if job j is assigned to the time point v_ℓ, which means that the requirements of job j must be satisfied by the resource supplies up to time

point v_ℓ. The mathematical program is

$$C^*_{\max} = \min \max_{v_\ell \in \mathcal{T}} \left(v_\ell + \sum_{j \in \mathcal{J}} \sum_{\nu=\ell}^{\tau} p_j x_{j\nu} \right) \tag{1}$$

s.t.

$$\sum_{j \in \mathcal{J}} \sum_{\nu=1}^{\ell} a_{ij} x_{j\nu} \leq b_{\ell i}, \quad v_\ell \in \mathcal{T},\ i \in \mathcal{R} \tag{2}$$

$$\sum_{\ell=1}^{\tau} x_{j\ell} = 1, \quad j \in \mathcal{J} \tag{3}$$

$$x_{j\ell} = 0, \quad j \in \mathcal{J},\ v_\ell \in \mathcal{T} \text{ such that } r_j > v_\ell \tag{4}$$

$$x_{j\ell} \in \{0,1\}, \quad j \in \mathcal{J},\ v_\ell \in \mathcal{T}. \tag{5}$$

The objective function expresses the completion time of the job finished last using the observation that there is a time point, either a release date of some job, or when some resource is supplied, from which the machine processes the jobs without idle times. Constraints (2) ensure that the jobs assigned to time points v_1 through v_ℓ use only the resources supplied up to time v_ℓ. Equations (3) ensure that all jobs are assigned to some time point. Finally, no job may be assigned to a time point before its release date by (4). Any feasible job assignment $\bar{x}$ gives rise to a set of schedules which differ only in the ordering of jobs assigned to the same time point v_ℓ.

Notice that in a feasible solution $\hat{x}$ of (1)-(5) there can be more than one jobs assigned to the same time point v_ℓ. We obtain a schedule of the jobs by putting them on the machine in the order of their assignment to the time points in $\mathcal{T}$. That is, first we schedule in any order without idle times the jobs with $\hat{x}_{j1} = 1$ from time v_1 on. Let C_1 be the completion time of these jobs. In a general step $\ell \geq 2$, we schedule the jobs with $\hat{x}_{j\ell} = 1$ in any order after $\max\{C_{\ell-1}, v_\ell\}$, and we denote by C_ℓ the completion time of the job finished last in this group. The schedule obtained in this way is feasible, and its makespan is the completion time of the job finished last, which is necessarily equal to the objective function value of solution $\hat{x}$. Let $C_{\max}(\hat{x})$ denote this value.

3 Non-approximability of $1|rm|C_{\max}$

In this section we will prove Theorem 1. To this end, we will reduce the APX-complete Vertex Cover Problem in Bounded-degree graphs (*VERTEX-COVER-B*) to $1|rm|C_{\max}$. In VERTEX-COVER-B, given a connected graph and a constant B, such that the degree of any vertex is bounded by B. One has to find a subset U of vertices of minimum cardinality such that each edge of the graph is adjacent to some vertex in U. The following result was shown by Papadimitriou and Yannakakis (1991):

Theorem 6 *For any constant $B \geq 4$, there is a constant $\delta > 0$, such that it is NP-hard to approximate VERTEX-COVER-B better than $1 + \delta$.*

The above statement was strengthened by Alimonti and Kann (2000) showing APX-completeness for $B = 3$.

We can state the following well-known observation.

Proposition 1 *In a connected graph on n vertices in which the maximum degree of any vertex is B, the size of the minimum vertex cover is at least $\lceil (n-1)/B \rceil$.*

Now we are ready to prove the main result of this section.

Proof of Theorem 1. We will transform instances of VERTEX-COVER-B to instances of $1|rm, q = 2|C_{\max}$ and show that if the latter problem admits a polynomial time $(1+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$, then VERTEX-COVER-B has a $(1 + (B + 1)\varepsilon)$-approximation algorithm for any $\varepsilon > 0$, which contradicts Theorem 6.

In the course of the transformation, we map a connected graph $G = (V, E)$ of maximum degree B with $n = |V|$ vertices and $m = |E|$ edges to a scheduling problem instance with n jobs and $r = m$ non-renewable resources, and $q = 2$ supply dates. For each vertex $v \in V$ of the graph, we define one job, denoted by J_v, of processing time 1, and for each edge $e \in E$ we define one resource R_e. The resource requirements of job J_v are determined by the edges adjacent to the corresponding vertex v, that is, if edge e is adjacent to v, then job J_v requires one unit of the resource R_e. Let $u_1 = 0$, $u_2 = n - 1$, $\tilde{b}_{1,e} = 1$, and $\tilde{b}_{2,e} = 1$ for each resource R_e, i.e., from each resource one unit is supplied at time 0, and one more unit at time u_2.

What does a schedule of minimum length represent in the graph? Observe that in any feasible schedule, in the time interval u_1 and u_2 no two jobs requiring the same resource may be scheduled, since the initial supply from each resource is 1 unit. Let I be the set of vertices indexing those jobs scheduled between u_1 and u_2. Since no two jobs J_v and J_w with $v \neq w \in I$ may require the same resource (since $\tilde{b}_{1,e} = 1$ for each resource R_e), nodes v and w are not adjacent in the graph. Hence, I is an independent set in G. But then the vertices of G indexing those jobs scheduled after u_2 constitute a vertex cover of G (since the complement of an independent set is a vertex cover in any graph). Since $u_2 = (n - 1)$, and the graph is connected, in a schedule of minimum length, the vertices of G indexing those jobs scheduled after u_2 constitute an optimal vertex cover of G. Let vc^* denote the size of a minimum vertex cover of G. Then, the optimum makespan is $C^*_{\max} = u_2 + vc^*$.

Now we claim that if there is an $(1+\varepsilon)$-approximation algorithm for $1|rm, q = 2|C_{\max}$, then there is a $(1 + (B + 1)\varepsilon)$-approximation algorithm for VERTEX-COVER-B. So suppose we have an $(1+\varepsilon)$-approximation algorithm for $1|rm, q = 2|C_{\max}$. The schedule supplied by this algorithm on the set of instances constructed above satisfies the following:

$$C_{\max} = (n-1) + k \leq ((n-1) + vc^*)(1+\varepsilon),$$

where k is the number of jobs scheduled after $u_2 = (n - 1)$. Notice that k is the size of the vertex cover in G determined by the jobs scheduled after u_2. After rearranging terms we get

$$k \leq (n-1)\varepsilon + vc^*(1+\varepsilon).$$

Finally, using Proposition 1 we obtain

$$k \leq \frac{(n-1)}{B}(\varepsilon B) + vc^*(1+\varepsilon) \leq vc^*(1+(B+1)\varepsilon).$$

That is, the schedule determines a vertex cover of size at most $vc^*(1+(B+1)\varepsilon)$.

Notice that in the above proof we have provided an L-reduction from VERTEX-COVER-B to $1|rm|C_{\max}$ with parameters $\alpha = (B+1)$, and $\beta = 1$.

Notice that the vertex-cover problem appears in completely different scheduling problems, see e.g. Ambühl et al. (2011).

4 PTAS for fixed number of supply dates, resources and release dates

In this section we describe a polynomial time approximation scheme for scheduling resource consuming jobs on a single machine and with a constant number of non-renewable resources such that the number of supply dates and also the number of distinct job release dates *before* u_q are bounded by constants, and the objective is to minimize the makespan, shortly for the problem $1|rm = const., q = const., \#\{r_j \ : \ r_j < u_q\} = const.|C_{\max}$. We will reuse some of the ideas from Györgyi and Kis (2014). That is, we will divide the set of jobs into big and small ones, and schedule them separately starting with the big ones.

Let $p_{\text{sum}} := \sum_{j\in\mathcal{J}} p_j$ denote the sum of the processing times of all the jobs. For a fixed $\varepsilon > 0$, let $\mathcal{B} := \{j \in \mathcal{J} \mid p_j \geq \varepsilon p_{\text{sum}}\}$ be the set of big jobs, and $\mathcal{S} := \mathcal{J} \setminus \mathcal{B}$ the set of small jobs. We divide further the set of small jobs according to their release dates, that is, we define the sets $\mathcal{S}^b := \{j \in \mathcal{S} \mid r_j < u_q\}$, and $\mathcal{S}^a := \mathcal{S} \setminus \mathcal{S}^b$. Let $\mathcal{T}^b := \{v_\ell \in \mathcal{T} \mid v_\ell < u_q\}$ be the set of time points v_ℓ before u_q. The following observation reduces the number of solutions of (1)-(5) to be examined.

Proposition 2 *From any feasible solution $\hat{x}$ of (1)-(5), we can obtain a solution $\tilde{x}$ with $C_{\max}(\tilde{x}) \leq C_{\max}(\hat{x})$ such that each job J_j is assigned to some time point v_ℓ ($\tilde{x}_{j\ell} = 1$), satisfying either $v_\ell < u_q$, or $v_\ell = \max\{u_q, r_j\}$.*

Proof Let $\mathcal{J}^a(\hat{x})$ be the subset of jobs with $\hat{x}_{j\ell} = 1$ for some $v_\ell > u_q$. We define a new solution $\tilde{x}$ in which those jobs in $\mathcal{J}^a(\hat{x})$ are reassigned to new time points and show that $C_{\max}(\tilde{x}) \leq C_{\max}(\hat{x})$. Let $\tilde{x} \in \{0,1\}^{\mathcal{J}\times\mathcal{T}}$ be a binary vector which agrees with $\hat{x}$ for those jobs in $\mathcal{J} \setminus \mathcal{J}^a(\hat{x})$. For each $j \in \mathcal{J}^a(\hat{x})$, let $\tilde{x}_{j\ell} = 1$ for $v_\ell = \max\{u_q, r_j\}$, and 0 otherwise. We claim that $\tilde{x}$ is a feasible solution of (1)-(5), and that $C_{\max}(\tilde{x}) \leq C_{\max}(\hat{x})$. Feasibility of $\tilde{x}$ follows from the fact that u_q is the last time point when some resource is supplied, and that no job is assigned to some time point before its release date. As for the second claim, consider the objective function (1). We will verify that for each $\ell = 1, \ldots, \tau$,

$$v_\ell + \sum_{j\in\mathcal{J}} \sum_{\nu=\ell}^{\tau} p_j \tilde{x}_{j\nu} \leq v_\ell + \sum_{j\in\mathcal{J}} \sum_{\nu=\ell}^{\tau} p_j \hat{x}_{j\nu}, \tag{6}$$

from which the claim follows. If $v_\ell \leq u_q$, the left and the right-hand sides in (6) are equal. Now consider any ℓ with $v_\ell > u_q$. Since no job in $\mathcal{J}^a(\hat{x})$ is assigned to a later time point in $\tilde{x}$ than in $\hat{x}$, the inequality (6) is verified again.

Notice that Proposition 2 also follows form a results of Lawler (1973) which implies that if we have a single machine and a set of jobs with release dates, and the objective is the minimum makespan, then it is optimal to sequence the jobs in non-decreasing release date order.

An *assignment of big jobs to the time points v_1 through v_τ* is given by a partial solution $x^{big} \in \{0,1\}^{\mathcal{B}\times\mathcal{T}}$ which assigns each big job to some time point v_ℓ. An assignment x^{big} of big jobs is *feasible* if the vector $x = (x^{big}, 0) \in \{0,1\}^{\mathcal{J}\times\mathcal{T}}$ satisfies (2), (4) and also (3) for the big jobs. Consider any feasible assignment x^{big} of big jobs. If we fix the assignment of the big jobs in (2)-(4) to x^{big}, then the supply from any resource i up to time point v_ℓ is decreased by the requirements of those big jobs assigned to time points v_1 through v_ℓ. Hence, we define the *residual resource supply* up to time point v_ℓ as $\bar{b}_{\ell i} := b_{\ell i} - \sum_{j\in\mathcal{B}} a_{ij}\left(\sum_{\mu=1}^{\ell} x_{j\mu}^{big}\right)$. Further on, let $\bar{C}_\ell^{\mathcal{B}} := \max_{\mu=1,\ldots,\ell}(v_\mu + \sum_{\kappa=\mu}^{\ell}\sum_{j\in\mathcal{B}} p_j x_{j\kappa}^{big})$ denote the earliest time point when the big jobs assigned to v_1 through v_ℓ may finish.

In order to assign approximately the small jobs, we will solve a linear program and round its solution. Our linear programming formulation relies on the following result.

Proposition 3 *There exists an optimal solution $(\hat{x}^{big}, \hat{x}^{small})$ of (1)-(5) such that for each $v_\ell \in \mathcal{T}^b$:*

$$\sum_{j\in\mathcal{S}^b} p_j \hat{x}_{j\ell}^{small} \le \max\{0, v_{\ell+1} - \bar{C}_\ell^{\mathcal{B}}\} + \varepsilon p_{\mathrm{sum}}. \tag{7}$$

Proof Suppose $(\hat{x}^{big}, \hat{x}^{small})$ is an optimal solution which does not meet the property claimed. Without loss of generality, we may assume that in the optimal schedule corresponding to $(\hat{x}^{big}, \hat{x}^{small})$, for each $v_k \in \mathcal{T}$, small jobs assigned to v_k follow the big ones assigned to v_k. Let $v_\ell \in \mathcal{T}^b$ be the smallest time point for which (7) is violated. Then some small jobs assigned to v_ℓ necessarily start after $v_{\ell+1}$ in any schedule corresponding to $(\hat{x}^{big}, \hat{x}^{small})$. Since all small jobs are of processing time less than $\varepsilon p_{\mathrm{sum}}$, we can reassign some of the small jobs from time point v_ℓ to $v_{\ell+1}$ until (7) is satisfied for v_ℓ. Clearly, such a reassignment of small jobs does not increase the length of the schedule. Then we proceed with the next time point in $\mathcal{T}$ until we get a schedule meeting (7).

Now, the linear program is defined with respect to any feasible assignment x^{big} of the big jobs:

$$\max \sum_{v_\ell \in \mathcal{T}^b} \sum_{j \in \mathcal{S}^b} p_j x_{j\ell}^{small} \tag{8}$$

s.t.

$$\sum_{j \in \mathcal{S}^b} \sum_{\nu=1}^{\ell} a_{ij} x_{j\nu}^{small} \leq \bar{b}_{\ell i}, \quad v_\ell \in \mathcal{T}^b,\ i \in \mathcal{R} \tag{9}$$

$$\sum_{j \in \mathcal{S}^b} p_j x_{j\ell}^{small} \leq \max\{0, v_{\ell+1} - \bar{C}_\ell^{\mathcal{B}}\} + \varepsilon p_{\text{sum}}, \quad v_\ell \in \mathcal{T}^b \tag{10}$$

$$\sum_{v_\ell \in \mathcal{T}^b \cup \{u_q\}} x_{j\ell}^{small} = 1, \quad j \in \mathcal{S}^b \tag{11}$$

$$x_{j\ell}^{small} = 0, \quad j \in \mathcal{S}^b,\ v_\ell \in \mathcal{T} \text{ such that } v_\ell < r_j, \text{ or } v_\ell > u_q \tag{12}$$

$$x_{j\ell}^{small} \geq 0, \quad j \in \mathcal{S}^b,\ v_\ell \in \mathcal{T}. \tag{13}$$

The objective function (8) maximizes the total processing time of those small jobs assigned to some time point v_ℓ *before* u_q. Constraints (9) make sure that no resource is overused taking into account the fixed assignment of big jobs as well. Inequalities (10) ensure that the small jobs assigned to v_ℓ fit into the interval $[\bar{C}_\ell^{\mathcal{B}}, v_{\ell+1} + \varepsilon p_{\text{sum}})$. Due to (11), small jobs are assigned to some time point in $\mathcal{T}^b \cup \{u_q\}$. The release dates of those jobs in $\mathcal{S}^b$, and Proposition 2 are taken care of by (12). Finally, we require that the values $x_{j\ell}^{small}$ be non-negative.

Notice that this linear program always has a finite optimum provided that x^{big} is a feasible assignment of the big jobs. Let $\bar{x}^{small}$ be any feasible solution of the linear program. Job $j \in \mathcal{S}^b$ is *integral* in $\bar{x}^{small}$ if there exists $v_\ell \in \mathcal{T}$ with $\bar{x}_{j\ell}^{small} = 1$, otherwise it is *fractional*. After all these preliminaries, the PTAS is as follows.

Algorithm A

1. Assign the big jobs to time points v_1 through v_τ in all possible ways which satisfies Proposition 2, and for each feasible assignment x^{big} do steps 2-5:
2. Define and solve linear program (8)-(13), and let $\bar{x}^{small}$ be an optimal basic solution.
3. Round each fractional value in $\bar{x}^{small}$ down to 0, and let $x^{small} := \lfloor \bar{x}^{small} \rfloor$ be the resulting partial assignment of small jobs, and $U \subset \mathcal{S}^b$ the set of fractional jobs in $\bar{x}^{small}$.
4. Finally, each $j \in U$ is assigned to time point $u_q \in \mathcal{T}$ by letting $x_{j\ell}^{small} := 1$ for all $j \in U$ where $v_\ell = u_q$, and all jobs in $\mathcal{S}^a$ are assigned to their release dates by setting $x_{j\ell}^{small} := 1$ for all $j \in \mathcal{S}^a$, where $v_\ell = r_j$ (recall that the release dates of jobs belong to set $\mathcal{T}$).
5. If the value of the complete assignment (x^{big}, x^{small}) of all the jobs is better than the best solution found so far, then update the best solution to (x^{big}, x^{small}).
6. After examining each feasible assignment of big jobs, output the best complete solution found.

We have to verify that the solution found by the above algorithm is feasible for (1)-(5), its value is not too far from the optimum, and that the algorithm runs in polynomial time in the size of the input. We introduce some terminology to facilitate the following discussion. A 0/1-vector (x^{big}, x^{small}) satisfying constraints (2) and (4) constitutes a *complete solution* if every job is assigned to some time point v_ℓ, i.e., it satisfies also the equations (3), otherwise it is a *partial solution.*

Lemma 1 *Every complete solution (x^{big}, x^{small}) constructed by the algorithm is feasible for (1)-(5).*

Proof Since the algorithm examines only feasible assignments of big jobs, $(x^{big}, 0)$ satisfies (2), (4), (5), and also (3) for all jobs in $\mathcal{B}$ by definition. The binary vector $(x^{big}, \lfloor \bar{x}^{small} \rfloor, 0)$ consists of the assignment of big jobs, and of those small jobs in $\mathcal{S}^b$ which are integral in the optimal solution $\bar{x}^{small}$ of the linear program. Notice that the fractional jobs in $\bar{x}^{small}$, and all jobs in $\mathcal{S}^a$ are unassigned. The partial solution $(x^{big}, \lfloor \bar{x}^{small} \rfloor, 0)$ satisfies (2), (4), (5), and also (3) for all jobs in $\mathcal{B} \cup \{j \in \mathcal{S}^b \mid j \text{ is integral in } \bar{x}^{small}\}$, since $\bar{x}^{small}$ is a feasible solution of (8)-(13). Finally, since u_q is the last time point when some resource is supplied and all job in $\mathcal{S}^a \cup U$ are assigned to some time points not before u_q, the 0/1-vector (x^{big}, x^{small}) is feasible for (1)-(5).

We will need the following result:

Proposition 4 *In any basic solution of the linear program (8)-(13), there are at most $(|\mathcal{R}| + 1) \cdot |\mathcal{T}^b|$ fractional jobs.*

Proof Let $\bar{x}^{small}$ be a basic solution of the linear program in which f jobs of $\mathcal{S}^b$ are assigned fractionally, and $e = |\mathcal{S}^b| - f$ jobs integrally. Clearly, each integral job gives rise to precisely one positive value, and each fractionally assigned job to at least two. This program has $|\mathcal{S}^b| \cdot |\mathcal{T}^b|$ decision variables, and $m = |\mathcal{S}^b| + (|\mathcal{R}| + 1) \cdot |\mathcal{T}^b|$ constraints. Therefore, in $\bar{x}^{small}$ there are at most m positive values, as no variable may be nonbasic with a positive value. Hence,

$$e + 2f \leq |\mathcal{S}^b| + (|\mathcal{R}| + 1) \cdot |\mathcal{T}^b| = e + f + (|\mathcal{R}| + 1) \cdot |\mathcal{T}^b|.$$

This implies

$$f \leq (|\mathcal{R}| + 1) \cdot |\mathcal{T}^b|$$

as claimed.

Lemma 2 *The algorithm constructs at least one complete assignment (x^{big}, x^{small}) whose value is at most $(1 + O(\varepsilon))$ times the optimum makespan $C^*_{\max}$.*

Proof Consider an optimal solution $(\hat{x}^{big}, \hat{x}^{small})$ of (1)-(5), consisting of the assignment of big jobs and that of the small jobs. We can suppose that this solution satisfies the condition of Proposition 2. The algorithm will examine $\hat{x}^{big}$, being a feasible assignment of the big jobs. Let $x^a \in \{0,1\}^{\mathcal{S}^a \times \mathcal{T}}$ be an assignment of the small jobs in $\mathcal{S}^a$ with $x^a_{j\ell} = 1$ if and only if $r_j = v_\ell$. Let $C_{\max}((\hat{x}^{big}, 0, x^a))$ be the value of the partial assignment $(\hat{x}^{big}, 0, x^a)$ in which no job in $\mathcal{S}^b$ is assigned to any time point. Clearly, $C_{\max}((\hat{x}^{big}, 0, x^a)) \leq C_{\max}((\hat{x}^{big}, \hat{x}^{small})) = C^*_{\max}$. Now,

let us consider the assignment of the small jobs in $\mathcal{S}^b$ constructed by the algorithm. Let $\lfloor \bar{x}^{small} \rfloor \in \{0,1\}^{\mathcal{S}^b \times (\mathcal{T}^b \cup \{u_q\})}$ be the assignment of the small jobs in $\mathcal{S}^b$ assigned by Algorithm A integrally. Let $C_{\max}((\hat{x}^{big}, \lfloor \bar{x}^{small} \rfloor, x^a))$ be the value of the partial assignment $(\hat{x}^{big}, \lfloor \bar{x}^{small} \rfloor, x^a)$.

Since $\bar{x}^{small}$ is a feasible solution of (8)-(13), the partial solution $(\hat{x}^{big}, \lfloor \bar{x}^{small} \rfloor, x^a)$ may assign small jobs of total processing time at most $\max\{0, v_{\ell+1} - \bar{C}_\ell^{\mathcal{B}}\} + \varepsilon p_{\text{sum}}$ to each $v_\ell \in \mathcal{T}^b$ (in addition to the big jobs assigned by x^{big}). Hence, the jobs assigned to time points $v_k \geq u_q$ may be pushed to the right by at most $|\mathcal{T}^b| \varepsilon p_{\text{sum}}$ by the small jobs assigned to time points in $\mathcal{T}^b$. Since the linear program maximizes $\sum_{v_\ell \in \mathcal{T}^b} \sum_{j \in \mathcal{S}^b} p_j x_{j\ell}$, thus $\sum_{j \in \mathcal{S}^b, v_\ell = u_q} p_j \bar{x}_{j\ell} \leq \sum_{j \in \mathcal{S}^b, v_\ell = u_q} p_j \hat{x}_{j\ell}$, therefore $C_{\max}((\hat{x}^{big}, \lfloor \bar{x}^{small} \rfloor, x^a)) \leq C_{\max}((\hat{x}^{big}, \hat{x}^{small})) + |\mathcal{T}^b| \varepsilon p_{\text{sum}} = C^*_{\max} + |\mathcal{T}^b| \varepsilon p_{\text{sum}}$.

Finally, we bound the total processing time of those jobs in set U. By Proposition 4, the size of U is at most $(|\mathcal{R}| + 1) \cdot |\mathcal{T}^b|$. But then, the total processing time of those small jobs from $\mathcal{S}^b$ that are moved to the end of the schedule is at most $(|\mathcal{R}| + 1) \cdot |\mathcal{T}^b| \cdot \varepsilon p_{\text{sum}}$. To summarize, we have

$$\begin{aligned} C_{\max}((\hat{x}^{big}, x^{small})) &\leq C_{\max}((\hat{x}^{big}, \lfloor \bar{x}^{small} \rfloor, x^a)) + (|\mathcal{R}| + 1) \cdot |\mathcal{T}^b| \cdot \varepsilon p_{\text{sum}} \\ &\leq C^*_{\max} + (|\mathcal{R}| + 2) \cdot |\mathcal{T}^b| \cdot \varepsilon p_{\text{sum}} \leq (1 + O(\varepsilon)) C^*_{\max}, \end{aligned}$$

where the first and the second inequality follows from the above discussion, and the last from the fact that both $|\mathcal{R}|$ and $|\mathcal{T}^b|$ are bounded by constants by assumption and from the observation that $p_{\text{sum}} \leq C^*_{\max}$.

Lemma 3 *For any fixed $\varepsilon > 0$, the running time of the algorithm is polynomial in the size of the input.*

Proof Since the processing time of each big job is at least $\varepsilon p_{\text{sum}}$, the number of big jobs is at most $\lfloor 1/\varepsilon \rfloor$, a constant, since ε is constant by assumption. Since the number of time points in $\mathcal{T}$ is also constant by assumption, the total number of assignments of big jobs to time point in $\mathcal{T}$ is also constant. For each feasible assignment, a linear program of polynomial size in the input must be solved. This can be accomplished by the Ellipsoid method in polynomial time (Gács and Lovász, 1981). Rounding the solution takes linear time in the number of small jobs. Hence, the entire running time is polynomial in the size of the input, as claimed.

Proof of Theorem 2. We show that Algorithm A is a PTAS for $1|rm = const., q = const., \#\{j : r_j < u_q\} = const.|C_{\max}$. The polynomial time complexity of the algorithm in the size of the input was shown in Lemma 3. In Lemma 2 it was shown that the performance ratio is $(1+O(\varepsilon))$, where the constant factor c in $O(\cdot)$ does not depend on the input. Hence, to reach a desired performance ratio δ, we let $\varepsilon := \delta/c$, and perform the computations with this choice of ε.

5 Arbitrary release dates

In this section we prove Theorem 3. We will reduce the problem $1|r_j, rm = const, q = const.|C_{\max}$ to the one with a constant number of release dates before u_q. To this end, for a fixed $\delta > 0$, we modify some of the job release dates to

define a new problem instance in which there are at most $1/\delta$ different job release dates before u_q. That is,

$$r'_j := \begin{cases} r_j & \text{if } r_j \geq u_q \\ g(r_j) \cdot \delta u_q & \text{if } r_j < u_q \end{cases}$$

where $g : \mathbb{Z}_+ \to \mathbb{Z}_+$, and $g(r_j)$ equals the smallest non-negative integer z such that $z \cdot \delta u_q \geq r_j$. Let $C^r_{\max}(x)$ denote the value of some solution x of the modified problem instance, and $C^{r^*}_{\max}$ the optimum makespan.

Lemma 4 $C^{r^*}_{\max} \leq C^*_{\max} + \delta u_q$.

Proof Consider an optimal schedule of the original problem, and let S^*_j denote the starting time of job $j \in \mathcal{J}$. Let $S'_j = S^*_j + \delta u_q$ for each $j \in \mathcal{J}$. We claim that S' is a feasible schedule for the modified problem with makespan $C^*_{\max} + \delta u_q$. Both claims follow from the following easy observation:

$$r'_j \leq r_j + \delta u_q \leq S^*_j + \delta u_q = S'_j, \ \forall j \in \mathcal{J}.$$

Apply the PTAS of Section 4 with error parameter δ to the modified problem instance to obtain a δ-approximate solution $\hat{x}^r$. We show that $\hat{x}^r$ is a 3δ-approximate solution for the original problem instance.

Proof of Theorem 3. We will show that for any $0 < \delta \leq 1$, $C_{\max}(\hat{x}^r) \leq (1 + 3\delta)C^*_{\max}$, and thus the above algorithm for $1|r_j, rm = const., q = const.|C_{\max}$ is a PTAS.

We have

$$C_{\max}(\hat{x}^r) \leq C^r_{\max}(\hat{x}^r) \leq (1+\delta)C^{r^*}_{\max} \leq (1+\delta)(C^*_{\max} + \delta u_q) \leq (1+3\delta)C^*_{\max},$$

where the first inequality follows from $r_j \leq r'_j$, the second from the fact that we have applied a PTAS with error parameter δ to the modified problem instance, the third form Lemma 4, and the last one from $u_q < C^*_{\max}$ (by Assumption 1), and from elementary calculations.

6 PTAS for the special case with $p_j = a_j$

In this section we prove Theorem 4. Notice that in the problem $1|rm = 1, p_j = a_j|C_{\max}$, the number of supply dates is part of the input, there is only a single resource, and the resource requirements of the jobs are proportional to their processing times.

Our algorithm is similar to that of Section 4, that is, we will divide the set of jobs into big and small ones, $\mathcal{B}$ and $\mathcal{S}$, and schedule the two subsets separately. Let $\varepsilon > 0$ be fixed. The set of big and small jobs are defined in the same ways as in Section 4. Let $\mathcal{T} = \{u_1, \ldots, u_q\}$. We assign the big jobs to time points in $\mathcal{T}$ in all possible ways. Notice that since the size of $\mathcal{B}$ is bounded by $1/\varepsilon$, which is a constant because $\varepsilon > 0$ is fixed, the number of big job assignments is polynomial in the size of the input. We will consider only *feasible assignments*, i.e., a binary vector $x^{big} \in \{0,1\}^{\mathcal{B}\times\mathcal{T}}$ is feasible if and only if it satisfies conditions (2), and (3) for $j \in \mathcal{B}$. (Notice that the condition (4) is void, since all job release dates are 0.)

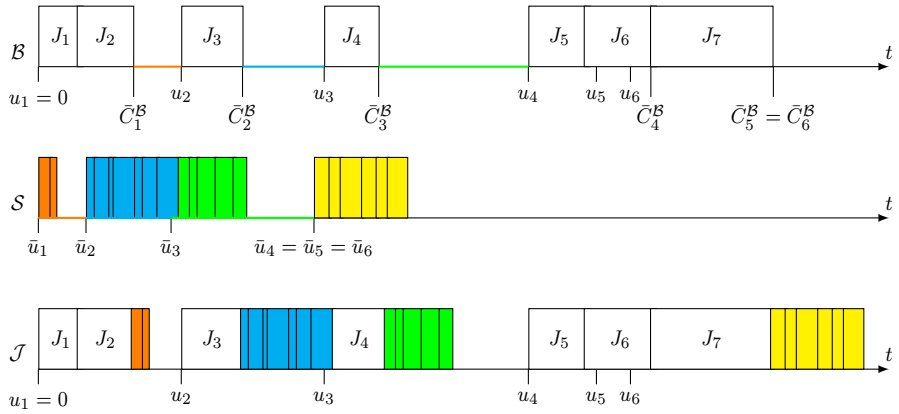

Fig. 1 Illustration of the algorithm

For each feasible assignment of big jobs, we assign the small jobs to time points in a suboptimal way. Finally, we choose the best complete assignment constructed. We will prove that the best solution found has a makespan of no more than $(1+\varepsilon)C^*_{\max}$, and that the algorithm has a polynomial time complexity.

For each feasible assignment x^{big} of the big jobs, we define an integer program for the remaining problem: let $\bar{C}_\ell^{\mathcal{B}}$ be the earliest completion time of the big jobs assigned to the first ℓ time points u_1 through u_ℓ, that is, $\bar{C}_1^{\mathcal{B}} := \sum_{j\in\mathcal{B}} p_j x_{j1}^{big}$, and for each $\ell \in \{2,\dots,q\}$, $\bar{C}_\ell^{\mathcal{B}} := \max\{\bar{C}_{\ell-1}^{\mathcal{B}}, u_\ell\} + \sum_{j\in\mathcal{B}} p_j x_{j\ell}^{big}$. Moreover, let $\bar{u}_\ell$ denote the total idle time until time point $\bar{C}_\ell^{\mathcal{B}}$ in the schedule of the big jobs, that is, $\bar{u}_\ell = \bar{C}_\ell^{\mathcal{B}} - \sum_{\nu=1}^{\ell}\sum_{j\in\mathcal{B}} p_j x_{j\nu}^{big}$, $\ell = 1,\dots,q$. Recall that $\bar{b}_\ell = b_\ell - \sum_{j\in\mathcal{B}} a_j \left(\sum_{\nu=1}^{\ell} x_{j\nu}^{big}\right) \geq 0$ is the residual resource supply. For a fixed feasible assignment of the big jobs, we define the following integer program.

$$\bar{C}^*_{\max} = \min \max_{\ell=1,\dots,q} \left(\bar{u}_\ell + \sum_{\nu=\ell}^{q} \sum_{j\in\mathcal{S}} p_j x_{j\nu}^{small} \right) \tag{14}$$

s.t.

$$\sum_{j\in\mathcal{S}} p_j \left(\sum_{\nu=1}^{\ell} x_{j\nu}^{small} \right) \leq \bar{b}_\ell, \quad \ell = 1,\dots,q \tag{15}$$

$$\sum_{\ell=1}^{q} x_{j\ell}^{small} = 1, \quad j \in \mathcal{S} \tag{16}$$

$$x_{j\ell}^{small} \in \{0,1\}, \quad \ell = 1,\dots,q,\ j \in \mathcal{S}. \tag{17}$$

The value of a feasible solution x^{small} will be denoted by $\bar{C}_{\max}(x^{small})$. After we have found a suboptimal solution of (14)-(17), we assign the small jobs to the u_ℓ according to this solution. See Figure 1 for an illustration of the whole algorithm.

The next statement follows from the definitions.

Proposition 5 *If x^{big} is a feasible assignment of the big jobs, then x^{small} is a feasible solution of (14)-(17) if and only if the complete solution (x^{big}, x^{small}) is a feasible solution of (1)-(5).*

For the sake of simpler presentation, we also define $\bar{u}_0 := 0$ and $\bar{C}_0^{\mathcal{B}} := 0$.

Proposition 6 *If x^{big} is a feasible assignment of the big jobs, then*

1. *if $\ell > 1$ and $\bar{u}_{\ell-1} < \bar{u}_\ell$, then $\bar{u}_\ell = \bar{u}_{\ell-1} + u_\ell - \bar{C}_{\ell-1}^{\mathcal{B}}$.*
2. *$\sum_{\nu=1}^{\ell-1} \sum_{j\in\mathcal{B}} p_j x_{j\nu}^{big} \geq u_\ell - \bar{u}_\ell, \quad \ell = 1, \ldots, q.$*

Proof Both claims follow from the definitions of $\bar{u}_\ell$ and $\bar{C}_\ell^{\mathcal{B}}$.

Proposition 7 *Consider the assignment $x = (x^{big}, x^{small})$, where x^{big} is an arbitrary feasible assignment of the big jobs, and x^{small} is an arbitrary assignment of the small jobs. Let ℓ_0 be the smallest index such that $\bar{C}_{\max}(x^{small}) = \bar{u}_{\ell_0} + \sum_{\nu=\ell_0}^{q} \sum_{j\in\mathcal{S}} p_j x_{j\nu}^{small}$. In this case $C_{\max}(x) = u_{\ell_0} + \sum_{\nu=\ell_0}^{q} \sum_{j\in\mathcal{J}} p_j x_{j\nu}$.*

Proof Since ℓ_0 is the smallest index with $\bar{C}_{\max}(x^{small}) = \bar{u}_{\ell_0} + \sum_{\nu=\ell_0}^{q} \sum_{j\in\mathcal{S}} p_j x_{j\nu}^{small}$, either $\ell_0 > 1$ and $\bar{u}_{\ell_0-1} < \bar{u}_{\ell_0}$, or $\ell_0 = 1$. In both cases $\bar{u}_{\ell_0} = \bar{u}_{\ell_0-1} + u_{\ell_0} - \bar{C}_{\ell_0-1}^{\mathcal{B}}$. By contradiction, suppose that there is an index $\ell \neq \ell_0$ such that

$$u_{\ell_0} + \sum_{\nu=\ell_0}^{q} \sum_{j\in\mathcal{J}} p_j x_{j\nu} < u_\ell + \sum_{\nu=\ell}^{q} \sum_{j\in\mathcal{J}} p_j x_{j\nu} = C_{\max}(x). \tag{18}$$

Choose the smallest index ℓ_1 among such indices. Suppose that $\ell_1 < \ell_0$ (the case $\ell_1 > \ell_0$ is similar). Rearranging (18) yields

$$u_{\ell_0} - u_{\ell_1} < \sum_{\nu=\ell_1}^{\ell_0-1} \sum_{j\in\mathcal{J}} p_j x_{j\nu}. \tag{19}$$

Observe that $\bar{u}_{\ell_0} + \sum_{\nu=\ell_0}^{q} \sum_{j\in\mathcal{S}} p_j x_{j\nu}^{small} > \bar{u}_{\ell_1} + \sum_{\nu=\ell_1}^{q} \sum_{j\in\mathcal{S}} p_j x_{j\nu}^{small}$ follows from the condition of the proposition, hence, $\bar{u}_{\ell_0} - \bar{u}_{\ell_1} > \sum_{\nu=\ell_1}^{\ell_0-1} \sum_{j\in\mathcal{S}} p_j x_{j\nu}^{small}$. Subtract it from (19) to get $u_{\ell_0} - u_{\ell_1} - \bar{u}_{\ell_0} + \bar{u}_{\ell_1} < \sum_{\nu=\ell_1}^{\ell_0-1} \sum_{j\in\mathcal{B}} p_j x_{j\nu}^{big}$. Since $\bar{u}_{\ell_0} = \bar{u}_{\ell_0-1} + u_{\ell_0} - \bar{C}_{\ell_0-1}^{\mathcal{B}} = u_{\ell_0} - \sum_{\nu=1}^{\ell_0-1} \sum_{j\in\mathcal{B}} p_j x_{j\nu}^{big}$ (by the definition of $\bar{u}_\ell$), thus $u_{\ell_1} - \bar{u}_{\ell_1} > \sum_{\nu=1}^{\ell_1-1} \sum_{j\in\mathcal{B}} p_j x_{j\nu}^{big}$ which contradicts Proposition 6, and the claim follows.

Proposition 8 *Consider a fixed assignment x^{big} of the big jobs, and two feasible solutions, x^{small} and $\tilde{x}^{small}$, of (14)-(17) such that $\bar{C}_{\max}(x^{small}) = \bar{C}_{\max}(\tilde{x}^{small}) + K$ for some $K \geq 0$. Then $C_{\max}((x^{big}, x^{small})) = C_{\max}((x^{big}, \tilde{x}^{small})) + K$.*

Proof Let ℓ be the smallest index such that $\bar{u}_\ell + \sum_{\nu=\ell}^{q} \sum_{j\in\mathcal{S}} p_j x_{j\nu}^{small} = \bar{C}_{\max}(x^{small})$. Since $\bar{u}_\ell > \bar{u}_{\ell-1}$ or $\ell = 1$, we have $\bar{u}_\ell = \bar{u}_{\ell-1} + u_\ell - \bar{C}_{\ell-1}^{\mathcal{B}}$. Using this and the definition of $\bar{u}_{\ell-1}$, we get $\bar{C}_{\max}(x^{small}) = u_\ell - \sum_{\nu=1}^{\ell-1} \sum_{j\in\mathcal{B}} p_j x_{j\nu}^{big} + \sum_{\nu=\ell}^{q} \sum_{j\in\mathcal{S}} p_j x_{j\nu}^{small}$. Define $\tilde{\ell}$ analogously for $\tilde{x}^{small}$, use the same transformation and plug in these formulas in the condition of the proposition:

$$u_\ell - \sum_{\nu=1}^{\ell-1} \sum_{j\in\mathcal{B}} p_j x_{j\nu}^{big} + \sum_{\nu=\ell}^{q} \sum_{j\in\mathcal{S}} p_j x_{j\nu}^{small} = u_{\tilde{\ell}} - \sum_{\nu=1}^{\tilde{\ell}-1} \sum_{j\in\mathcal{B}} p_j x_{j\nu}^{big} + \sum_{\nu=\tilde{\ell}}^{q} \sum_{j\in\mathcal{S}} p_j \tilde{x}_{j\nu}^{small} + K.$$

Add $\sum_{j\in\mathcal{B}} p_j$ to both sides and the proposition follows from Proposition 7.

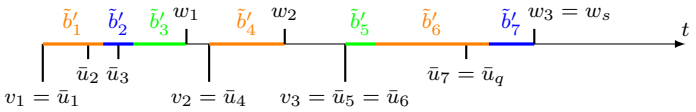

Fig. 2 Dividing the time horizon into intervals

We present a method to obtain a suboptimal solution of (14)-(17). Let $\tilde{b}'_1 := \bar{b}_1$ and $\tilde{b}'_\ell := \bar{b}_\ell - \bar{b}_{\ell-1},\ \ell = 2, \dots, q$ be the amount of the resource that becomes available for small jobs scheduled after $\bar{u}_\ell$. Note that the problem of assigning small jobs to time points is equivalent to the problem instance of $1|rm = 1, p_j = a_j|C_{\max}$ in which $\tilde{b}'_\ell$ amount of resource is supplied at $\bar{u}_\ell$, $\ell = 1, \dots, q$.

Since $p_j = a_j$, passing of time, and resource utilization is interchangeable, which motivates the following construction. We divide the time horizon into intervals $[v_\ell, w_\ell]$ in which the material supply is contiguous:

1. The first interval starts at time $v_1 := \bar{u}_1$, and let $\ell := 1$.
2. For $k = 2, \dots, q$ repeat the following step:
3. If $\bar{u}_k > v_\ell + \sum_{z:v_\ell \le \bar{u}_z < \bar{u}_k} \tilde{b}'_z$, then the end of the ℓ^{th} interval is $w_\ell := v_\ell + \sum_{z:v_\ell \le \bar{u}_z < \bar{u}_k} \tilde{b}'_z$, i.e., v_ℓ plus the amount supplied after v_ℓ and before u_k. Let $v_{\ell+1} := \bar{u}_k$, $\ell := \ell + 1$. Otherwise, proceed with the next k.
4. Let $s := \ell$, and $w_s := v_s + \sum_{z:v_s \le \bar{u}_z} \tilde{b}'_z$.

Notice that at the end of the algorithm, $1 \le s \le q$, each v_ℓ is equal to one of the $\bar{u}_k$ values, and the interval $[v_\ell, w_\ell]$ contains all the $\bar{u}_k$ values such that $\bar{u}_k \le v_\ell + \sum_{z:v_\ell \le \bar{u}_z < \bar{u}_k} \tilde{b}'_z$, for $\ell = 1, \dots, s$, that is, in each interval $[v_\ell, w_\ell]$, the next supply arrives before the previous supplies could be fully consumed, if the rate of consumption were constant 1. See Figure 2 for an illustration.

Proposition 9 *The optimal makespan of problem (14)-(17) is at least w_s.*

Proof The total resource supply before v_s is

$$\sum_{z:\bar{u}_z < v_s} \tilde{b}'_z = \sum_{\ell=1}^{q} \tilde{b}'_\ell - \sum_{z:v_s \le \bar{u}_z} \tilde{b}'_z = \sum_{j \in \mathcal{S}} a_j - \sum_{z:v_s \le \bar{u}_z} \tilde{b}'_z = \sum_{j \in \mathcal{S}} p_j - \sum_{z:v_s \le \bar{u}_z} \tilde{b}'_z,$$

thus the total amount of work scheduled before v_s is at most $\sum_{j \in \mathcal{S}} p_j - \sum_{z:v_s \le \bar{u}_z} \tilde{b}'_z$. Therefore every feasible schedule has to process at least $\sum_{z:v_s \le \bar{u}_z} \tilde{b}'_z$ amount of total work after v_s, and the proposition follows.

We will schedule the jobs to start in some interval, or after the end of $[v_s, w_s]$, when already all the materials are supplied. Let $\mathcal{I} := \bigcup_{\ell=1}^{s} [v_\ell, w_\ell]$. We define the function $F : \mathbb{Z}_+ \to \mathbb{Z}_+$ which provides for any $\mu \in \mathbb{Z}_+$ the earliest time point $t \in \mathcal{I}$ such that the measure of the set $[v_1, t] \cap \mathcal{I}$ is μ, i.e., $\int_{x \in [v_1,t] \cap \mathcal{I}} 1 dx = \mu$. If μ is too big, i.e., larger than the total size of the intervals, we let $F(\mu) := \infty$. Notice that F is monotone non-decreasing and piecewise linear. Let $p^S_{\max} := \max_{j \in \mathcal{S}} p_j$.

Algorithm B

1) Take an arbitrary order of the small jobs $(J_1, \dots, J_{|\mathcal{S}|})$.

2) For $j = 1, \ldots, |\mathcal{S}|$ repeat the following two steps:
3) Let $t := F(\sum_{j'=1}^{j-1} p_{j'} + p^S_{\max})$.
4) If $t < \infty$, then schedule job J_j at time t. Otherwise schedule J_j at $\max\{S_{j-1} + p_{j-1}, w_s\}$ if $j \geq 2$, or at w_s if $j = 1$.

Proposition 10 *The schedule constructed by Algorithm B is feasible, and it finishes at most $p^S_{\max}$ time units after w_s.*

Proof To verify feasibility, we have to check that (i) no job J_j starts before the previous job, if any, is finished, and that (ii) there is enough resource available to cover its demand. Property (i) follows from the monotonicity and piecewise linearity of F. As for (ii), if t is finite in step 3) of the algorithm, then by the definition of function F, the total supply in the intervals up to time t minus the total demand of the first $j-1$ jobs is at least $p^S_{\max}$. But, $p_j \leq p^S_{\max}$, and $a_j = p_j$, hence, there is enough resource available to start the job. If t is infinite, then already all the supplies from the resource arrived, and the job can certainly be started.

Finally, to see that the last job finishes at most $p^S_{\max}$ time units after w_s, observe that by the definition of F and by the construction of the schedule, the total resource available at time w_s is at most $p^S_{\max}$. Since $p_j = a_j$ for all jobs, and the total request of the small jobs equals $F(w_s)$, the total size of unscheduled jobs is at most $p^S_{\max}$.

The following algorithm creates a complete assignment of the jobs:

Algorithm C:

1. Assign the big jobs to time points u_1 through u_q in all possible ways, and for each feasible assignment x^{big} do steps 2-4.
2. Define the scheduling problem instance of $1|rm = 1, p_j = a_j|C_{\max}$ in which $\tilde{b}'_\ell$ amount of the resource is supplied at $\bar{u}_\ell$.
3. Construct a schedule according to Algorithm B, and let $x^{small}_{j\ell} := 1$ if and only if the starting time S_j of job $J_j \in \mathcal{S}$ satisfies $\bar{u}_\ell \leq S_j < \bar{u}_{\ell+1}, \quad \ell = 1, \ldots, q$, where $\bar{u}_{q+1} := \infty$.
4. If the value of the assignment $x = (x^{big}, x^{small})$ is better than the best solution found so far, then update the best solution to x.
5. Output the best solution found, denoted by x^C.

The next lemma follow from Propositions 9 and 10.

Lemma 5 *For every feasible assignment of the big jobs, the algorithm provides a small job assignment x^{small} such that $\bar{C}_{\max}(x^{small}) \leq \bar{C}^*_{\max} + p^S_{\max}$.*

Lemma 6 *The algorithm examines at least one feasible assignment (x^{big}, x^{small}) whose value is at most $p^S_{\max} \leq \varepsilon p_{\text{sum}}$ more than the optimum of (1)-(5).*

Proof Consider an optimal solution $(\hat{x}^{big}, \hat{x}^{small})$ of (1)-(5) and consider the case when $x^{big} = \hat{x}^{big}$. According to Proposition 10 and Proposition 5, the assignment (x^{big}, x^{small}) is feasible. The statement about the value of (x^{big}, x^{small}) follows from Lemma 5 and Proposition 8.

Proof of Theorem 4. Lemma 6 shows that the value of the solution x^C provided by Algorithm C is at most $\varepsilon p_{\text{sum}}$ more than the optimum makespan $C^*_{\max}$. Since $p_{\text{sum}} \leq C^*_{\max}$, it follows that $C_{\max}(x^C) \leq (1+\varepsilon)C^*_{\max}$. The running time is determined by the number of ways in which the big jobs can be assigned to the time points in $\mathcal{T}$, and for each such assignment the time needed to schedule the small jobs. The latter is clearly polynomial in the size of the problem instance with the small jobs. Since the number of big jobs is bounded by $1/\varepsilon$, which is a constant for any fixed $\varepsilon > 0$, we have shown that Algorithm C is a PTAS for $1|rm = 1, p_j = a_j|C_{\max}$.

7 A PTAS for the special case with $p_j = a_j$ and with arbitrary job release dates

In this section we sketch the proof of Theorem 5. Algorithm C can be easily extended to the case when the jobs have arbitrary release dates. We have to make the following modifications:

- Define $\mathcal{T} = \{u_\ell\}_{\ell=1}^{q} \cup \{r_j\}_{j\in\mathcal{J}} = \{v_1, \ldots, v_\tau\}$, like in Section 4.
- The feasible big job assignments have to satisfy (2), and (4).
- Determine the values $\bar{v}_1$ through $\bar{v}_\tau$ similarly as the values $\bar{u}_1$ through $\bar{u}_q$ in Section 6.
- In step 1 of Algorithm B, we have to process the small jobs in non-decreasing release date order.
- For each small job $j \in \mathcal{S}$, define $\bar{r}_j = \bar{v}_\ell$, where $v_\ell \in \mathcal{T}$ equals the release date r_j.
- In step 3 of Algorithm B, let $t := \max\{F(\sum_{j'=1}^{j-1} p_{j'} + p^S_{\max}), \max_{j_0 \leq j}\{\bar{r}_{j_0} + \sum_{j'=j_0}^{j-1} p_{j'}\}\}$, where the first part of the maximum ensures that there is enough resource available to start job j, while the second part ensures that the starting time of the job is not before its adjusted release date $\bar{r}_j$, and that the jobs do not overlap in time.

According to the new definition of the schedule it is easy to see the feasibility of the assignment. Again, we can prove that the obtained schedule has a makespan at most $p^S_{\max}$ greater than the optimal makespan, thus a statement analogous to Lemma 6 follows.

Finally, the running time is still polynomial in the size of the input, because the number of big job assignments is $O(\tau^{1/\varepsilon})$, which is polynomial in the size of the input, and the small jobs can also be scheduled by Algorithm B in polynomial time. This proves that Algorithm C with the above modifications is a PTAS for $1|rm = 1, p_j = a_j, r_j|C_{\max}$, thus Theorem 5 follows.

8 Conclusion

In this paper we have given new negative and positive results on the approximability of single machine scheduling with resource consuming jobs. One of the main open questions is whether the problem with $q = 3$ supply dates and a single non-renewable resource only admits a fully polynomial time approximation scheme.

Acknowledgments

The authors are indebted to a referee for careful reading and pointing out several errors in an earlier version of the paper. This work has been supported by the OTKA grant K112881. The research of Tamás Kis has been supported by the János Bolyai research grant BO/00412/12/3 of the Hungarian Academy of Sciences.

Conflict of Interest

The authors declare that they have no conflict of interest.

References

Alimonti P., Kann V. (2000): Some APX-completeness results for cubic graphs, *Theoretical Computer Science*, 237, 123–134.

Ambühl C.,Mastrolilli M., Mutsanas N., Svensson O. (2011): On the approximability of single-machine scheduling with precedence constraints, *Mathematics of Operations Research*, 36, 653–669.

Briskorn D., Choi B.-C., Lee K., Leung J., Pinedo M. (2010): Complexity of single machine scheduling subject to nonnegative inventory constraints, *European Journal of Operational Research*, 207, 605–619.

Briskorn D., Jaehn F., Pesch E. (2013): Exact algorithms for inventory constrained scheduling on a single machine, *Journal of Scheduling*, 16, 105–115.

Carlier J. (1984): Problèmes d'ordonnancements à contraintes de ressources: algorithmes et complexité, Thèse d'état

Carlier J., Rinnooy Kan A.H.G. (1982): Scheduling subject to nonrenewable resource constraints, *Operational Research Letters*, 1, 52–55.

Fleischer L., Goemans M.X., Mirrokni V.S., and Sviridenko M. (2011): Tight Approximation Algorithms for Maximum Separable Assignment Problems. *Mathematics of Operations Research* 36, 416–431.

Gács P., Lovász L. (1981): Khachiyan's algorithm for linear programming, *Mathematical Programming Studies*, 14, 61–68.

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, 7–13.

Gafarov E.R., Lazarev A.A., Werner F. (2014): Approximability results for the resource-constrained project scheduling problem with a single type of resources, *Annals of Operations Research*, 213, Issue 1, 115–130.

Garey M.R., Johnson D.S. (1979): *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco

Grigoriev A., Holthuijsen M., van de Klundert J. (2005): Basic scheduling problems with raw material constraints, *Naval Research of Logistics*, 52, 527–553.

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

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

P. Laborie (2003): Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results, *Artificial Intelligence* 143, 151–188.

Lawler E.L. (1973): Optimal sequencing of a single machine subject to precedence constraints, *Management Sci.* 19, 544–546.

Morsy E., Pesch E. (2015): Approximation algorithms for inventory constrained scheduling on a single machine on the same problem, *Journal of Scheduling*, to appear

Neumann K., Schwindt C. (2002): Project scheduling with inventory constraints, *Mathematical Methods of Operations Research*, 56, 513–533.

Papadimitriou C., Yannakakis M. (1991): Optimization, Approximation, and Complexity Classes, *Journal of Computer and System Sciences*, 43, 425–440.

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

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