Approximation schemes for parallel machine scheduling with non-renewable resources

Péter Györgyi[a,b], Tamás Kis[b,*]

[a]Department of Operations Research, Loránd Eötvös University, H1117 Budapest, Pázmány Péter sétány 1/C, Hungary
[b]Institute for Computer Science and Control, H1111 Budapest, Kende str. 13–17, Hungary

Abstract

In this paper the approximability of parallel machine scheduling problems with resource consuming jobs is studied. In these problems, in addition to a parallel machine environment, there are non-renewable resources, like raw materials, energy, or money, consumed by the jobs. Each resource has an initial stock, and some additional supplies at a-priori known moments in time and in known quantities. The schedules must respect the resource constraints as well. The optimization objective is either the makespan, or the maximum lateness. Polynomial time approximation schemes are provided under various assumptions, and it is shown that the makespan minimization problem is APX-complete if the number of machines is part of the input even if there are only two resources.

Keywords: Scheduling, parallel machines, non-renewable resources, approximation schemes

1. Introduction

In Supply Chains, non-renwable resources like raw materials, or energy are taken into account from the design through the operational levels. Advanced planning systems explicitly model and optmize their usage at various planning levels, see e.g., Chapters 4, 9 and 10 of Stadtler & Kilger (2008). In this paper,

* Corresponding author
Email addresses: gyorgyi.peter@sztaki.mta.hu (Péter Györgyi), kis.tamas@sztaki.mta.hu (Tamás Kis)

we focus on short-term scheduling, where in addition to machines, there are non-renewable resources 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.

More formally, there are m parallel machines, $\mathcal{M} = \{M_1, \ldots, M_m\}$, a finite set of n jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$, and a finite set of non-renewable resources $\mathcal{R}$ consumed by the jobs. 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 $i \in \mathcal{R}$. Preemption of jobs is not allowed and each machine can process at most one job at a time. The resources are supplied in q different time moments, $0 = u_1 < u_2 < \ldots < u_q$; the vector $\tilde{b}_\ell \in \mathbb{Z}_+^{|\mathcal{R}|}$ represents the quantities supplied at u_ℓ. A *schedule* σ specifies a machine and the starting time S_j of each job and it is *feasible* if (i) on every machine the jobs do not overlap in time, (ii) $S_j \geq r_j$ for each $j \in \mathcal{J}$, and if (iii) at any time point t the total supply from each resource is at least the total request of those jobs starting not later than t, i.e., $\sum_{(\ell\,:\,u_\ell \leq t)} \tilde{b}_{\ell i} \geq \sum_{(j\,:\,S_j \leq t)} a_{ij}$, $\forall i \in \mathcal{R}$. We will consider two types of objective functions: the minimization of the maximum job completion time (makespan) defined by $C_{\max} = \max_{j \in J} C_j$; and the minimization of the maximum lateness, i.e., each job has a due-date d_j, $j \in \mathcal{J}$, and $L_{\max} := \max_{j \in \mathcal{J}} (C_j - d_j)$. Clearly, $L_{\max}$ is a generalization of $C_{\max}$.

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.*

Since the makespan minimization problem with resource consuming jobs on a single machine is NP-hard even if there are only two supply dates (Carlier, 1984), all problems studied in this paper are NP-hard.

Scheduling with non-renewable resources has a great practical interest. Chapter 4 of (Stadtler & Kilger, 2008) describes examples in consumer goods industry and in computer assembly, where purchased items have to be taken into account at several planning levels including short-term scheduling which is the topic of the present paper. Herr & Goel (2016) study a scheduling problem arising in the continuous casting stage of steel production. A continuous caster is fed with ladles of liquid steel, where each ladle contains a certain steel grade and has orders allocated to it that determine a due date. The liquid steel is produced from hot iron supplied by a blast furnace with a constant rate. The sequence of ladles, including setups between ladles of different setup families, is not allowed to consume more hot metal than supplied by the blast furnace. Belkaid et al. (2012) study a problem of order picking in a platform with a distribution company that leads to the model considered in this paper. In Carrera et al. (2010), a similar problem is investigated in a shoe-firm. Further applications can be found in Section 2.

In this paper we take a theoretical viewpoint and analyze the approximability of parallel machine scheduling problems augmented with non-renewable resources. We believe that our study leads to a deeper understanding of the problem, that may facilitate the development of efficient practical algorithms.

1.1. Terminology

An *optimization problem* Π consists of a set of instances, where each instance has a set of *feasible solutions*, and each solution has an (objective function) value. In a *minimization problem* a feasible solution of minimum value is sought, while in a maximization problem one of maximum value. 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.

Observation 1. *For a PTAS for some problem Π, it is sufficient to provide a family of algorithms $\{A_\varepsilon\}_{\varepsilon>0}$ where each A_ε is an $c\cdot\varepsilon$-approximation algorithm for Π, where the constant factor c does not depend on the input or on ε. Then, letting $\varepsilon := \delta/c$, we get a PTAS $\{A_{(\delta/c)}\}_{\delta>0}$ for Π.*

We use the standard $\alpha|\beta|\gamma$ notation for scheduling problems (Graham et al. (1979)), where α denotes the processing environment, β the additional restrictions, and γ the objective function. In this paper, $\alpha = Pm$, which indicates m parallel machines for some fixed m. In the β field, 'rm' means that there are non-renewable resource constraints, $rm = r$ indicates $|\mathcal{R}| = r$. Further options are $q = const$ meaning that the number of supplies is a fixed constant, r_j indicates job release dates, while the restriction $\#\{r_j : r_j < u_q\} \leq const$ bounds the number of distinct job release dates before the last supply date u_q by a constant. For a set H, we define $p(H) := \sum_{j \in H} p_j$.

Throughout the paper we will consider *monotone* objective functions $F_{\max}$ that satisfy the following conditions:

(i) $F_{\max}$ is monotone increasing in the job completion times, i.e., $F_{\max}(C_1, \ldots, C_n) \leq F_{\max}(C'_1, \ldots, C'_n)$, for arbitrary $0 \leq C_j \leq C'_j$, $j = 1, \ldots, n$,

(ii) Its value does not grow faster than the value of any of its arguments, i.e., $F_{\max}(C_1 + \delta, \ldots, C_n + \delta) \leq F_{\max}(C_1, \ldots, C_n) + \delta$ for any $\delta \geq 0$,

(iii) On any instance, and for any feasible schedule, $F_{\max}$ is at least u_q.

Notice that e.g., the makespan, and the maximum lateness increased by some (instance dependent) constant satisfy the above properties, but the total completion time does not. From now on $F_{\max}$ denotes an arbitrary monotone objective function.

1.2. Main results

If the number of the machines is part of the input, then we have the following non-approximability result:

Theorem 1. *Deciding whether there is a schedule of makespan 2 with two non-renewable resources, two supply dates and unit-time jobs on an arbitrary number of machines ($P|rm = 2, q = 2, p_j = 1|C_{\max} \leq 2$) is NP-hard.*

Corollary 1. *It is NP-hard to approximate problem $P|rm = 2, q = 2, p_j = 1|C_{\max} \leq 2$ better than $3/2 - \varepsilon$ for any $\varepsilon > 0$.*

Obj.	#Machines m	#Supplies q	#Resources rm	Release dates r_j	PTAS	FPTAS
	1	2	1	no	yes[b]	yes[bc]
	1	2	1	yes	yes[d]	?
	1	2	$const. \geq 2$	yes/no	yes[cd]	no[c]
	1	2	arbitrary	yes/no	no[d]	no[c]
$C_{\max}$	1	$const. \geq 3$	1	yes/no	yes[bd]	?
	1	$const. \geq 3$	$const. \geq 2$	yes/no	yes[d]	no[c]
	1	arbitrary	1[*]	yes/no	yes[d]	no[a]
	$const \geq 2$ +ddc. jobs[**]	arbitrary	$const. \geq 2$	yes/no	yes (Sect. 5) yes (Sect. 6)	no[a]
	arbitrary	2	2	yes/no	no (Sect. 3)	no[e]
	arbitrary	arbitrary	1	yes/no	?	no[e]
$L'_{\max}$	*const*	arbitrary	1[*]	no	yes (Sect. 7)	?
$\sum w_j C_j$	1	2	1	no	yes[f]	yes[f]

[*] under the condition $a_j = \lambda p_j$ [**] even if only a $\mathcal{J}' \subseteq \mathcal{J}$ subset of jobs is dedicated

[a] Grigoriev et al. (2005) [b] Györgyi & Kis (2014) [c] Györgyi & Kis (2015a)

[d] Györgyi & Kis (2015b) [e] Garey & Johnson (1979) [f] Kis (2015)

Table 1: Known approximability results for scheduling problems with resource consuming jobs if $\mathcal{P} \neq \mathcal{NP}$. 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.

By assumption 1, the optimum makespan is at least u_q, therefore, a straightforward two-approximation algorithm would schedule all the jobs after u_q. Therefore, we have the following result.

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

The following result helps to obtain polynomial time approximation schemes for the general problem $P[m]|rm, r_j|F_{\max}$, provided that we have a family of approximation algorithms for restricted versions of the problem.

Proposition 1. *In order to have a PTAS for $P[m]|rm, r_j|F_{\max}$, it suffices to provide a family of algorithms $\{A_\varepsilon\}_{\varepsilon>0}$ such that A_ε is an ε-approximation algorithm for the restricted problem where the supply dates and the job release dates before u_q are from the set $\{\ell\varepsilon u_q : \ell = 0, 1, 2, \dots, \lfloor 1/\varepsilon \rfloor\}$.*

Using Proposition 1, we can prove the following result:

Theorem 2. *$Pm|rm = const., r_j|C_{\max}$ admits a PTAS.*

Notice that a PTAS has been known only for $1|rm = const, q = const, \#\{r_j : r_j < u_q\} \leq const|C_{\max}$ (Györgyi & Kis, 2015b). If the jobs are dedicated to machines, we have an analogous statement:

Theorem 3. *$Pm|rm = const., r_j, ddc|C_{\max}$ admits a PTAS.*

Now we turn to the $L_{\max}$ objective. Since the optimum lateness may be 0 or negative, a standard trick is to increase the lateness of the jobs by a constant that depends on the input. In our case, let $L'_{\max} := \max_j\{C_j - d_j + D\}$, where $D := \max_{j\in\mathcal{J}}\{d_j\} + u_q$. Note that this function satisfies the conditions (i)-(iii), thus it is a monotone objective function. In order to provide a PTAS for the lateness objective, we have to assume that the processing times are proportional to the resource consumptions. Such a model with the makespan objective has already been studied in (Györgyi & Kis, 2015b).

Theorem 4. *If $L'_{\max}$ is defined as above, then $Pm|rm = 1, p_j = a_j|L'_{\max}$ admits a PTAS.*

In Table 1 we summarize known and new approximability results for scheduling resource consuming jobs in single machine as well as in parallel machine environments, when preemption of processing is not allowed, and the resources are consumed right at starting the jobs. The table contains results for the makespan, the maximum lateness, and the weighted completion time objectives. These results complement the large body of approximation algorithms for NP-hard single and parallel machine scheduling problems (Williamson & Shmoys, 2011).

1.3. Structure of the paper

In Section 2 we summarize previous work on machine scheduling with non-renewable resources. In Section 3 we prove our hardness result Theorem 1. Then in Section 4 we establish Proposition 1. In Sections 5, 6, and 7 we prove Theorems 2, 3, and 4, respectively. Finally, we conclude the paper in Section 8.

2. Previous work

Scheduling problems with resource consuming jobs were introduced by Carlier (1984), Carlier & Rinnooy Kan (1982), and Slowinski (1984). In Carlier (1984), the computational complexity of several variants with a single machine was established, while in Carlier & Rinnooy Kan (1982) activity networks requiring only non-renewable resources were considered. In Slowinski (1984) a parallel machine problem with preemptive jobs was studied, and the single non-renewable resource had an initial stock and some additional supplies, like in the model presented above, and it was assumed that the rate of consuming the non-renewable resource was constant during the execution of the jobs. These assumptions led to a polynomial time algorithm for minimizing the makespan, which is in strong contrast to the NP-hardness of all the scheduling problems analyzed in this paper. Further results can be found in e.g., Toker et al. (1991), Xie (1997), Neumann & Schwindt (2003), Laborie (2003), Grigoriev et al. (2005), Briskorn et al. (2010), Briskorn et al. (2013), Gafarov et al. (2011), Györgyi & Kis (2014), Györgyi & Kis (2015a), Györgyi & Kis (2015b), Morsy & Pesch (2015). In particular, Toker et al. (1991) proved that scheduling jobs requiring one non-renewable resource on a single machine with the objective of minimizing the makespan reduces to the 2-machine flow shop problem provided that the single non-renewable resource has a unit supply in every time period. Neumann & Schwindt (2003) study general project scheduling problems with inventory constraints, and propose a branch-and-bound algorithm for minimizing the project length. In a more general setting, jobs may consume as well as produce non-renewable resources. In Xie (1997), 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 & 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) scheduling problems with the objective of minimizing the inventory levels were studied. Morsy & Pesch (2015) designed approximation algorithms to minimize the total weighted completion time. In Györgyi & 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 & Kis (2015a) 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 & Kis, 2014). These results have been extended in Györgyi & Kis (2015b): it contains a PTAS under various assumptions: (1) both the number of resources and the number of supplies dates are constants, (2) there is only one resource, an arbitrary number of supply dates, but the resource requirements are proportional to job processing times. It also proves the APX-hardness of the problem when the number of resources is part of the input.

Since the parallel machine environment can be considered as a renewable resource constraint (each job requires 1 unit during its proceeding, and there are m available units from this resource at each moment of time) our problem is a special case of the well-studied resource-constrained project scheduling problem. This problem has several practical application, e.g. the Process Move Programming Problem where, as in our problem, there are parallel machines and non-renewable resource constraints (Sirdey et al. (2007)). In many papers the resources can reduce the processing times, e.g., Shabtay & Kaspi (2006) deals with parallel machine problems with a non-renewable resource, while Janiak et al. (2007) provides a survey of that topic. Yeh et al. (2015) examined heuristic algorithms for a uniform parallel machine problem with resource consumption. Further theoretical and practical applications of the resource-constrained project scheduling can be found in Artigues et al. (2013).

3. APX-hardness of $P|rm = 2, q = 2, p_j = 1|C_{\max}$

In this section we prove Theorem 1. We reduce the EVEN-PARTITION problem to the problem $P|rm = 2, q = 2, p_j = 1|C_{\max}$, and argue that deciding whether a schedule of makespan two exists is as hard as finding a solution for EVEN-PARTITION. Recall that an instance of the EVEN-PARTITION problem consists of $2t$ items, for some integer t, of sizes $a_1, \dots, a_{2t} \in \mathbb{Z}_+$. The decision problem asks whether the set of items can be partitioned into two subsets S and $\bar{S}$ of cardinality t each, such that $\sum_{i\in S} a_i = \sum_{i\in \bar{S}} a_i$? This problem is NP-hard in the ordinary sense, see Garey & Johnson (1979). Clearly, a necessary condition for the existence of set S is that the total size of all items is an even integer, i.e., $\sum_{i=1}^{2t} a_i = 2A$, for some $A \in \mathbb{Z}_+$.

Proof of Theorem 1 We map an instance I of EVEN-PARTITION to the following instance of $P|rm = 2, q = 2, p_j = 1|C_{\max}$. There are $n := 2t$ jobs, and $m := t$ machines. All the jobs have unit processing time, i.e., $p_j = 1$ for all j. The job corresponding to the jth item in I has resource requirements $a_{1,j} := a_j$ and $a_{2,j} := A - a_j$. The initial supply at $u_1 = 0$ from the two resources is $\tilde{b}_{1,1} := A$ and $\tilde{b}_{1,2} := (t-1)A$, and the second supply at time $u_2 = 1$ has $\tilde{b}_{2,1} := A$, and $\tilde{b}_{2,2} := (t-1)A$. We have to decide whether a feasible schedule of makespan two exists.

First, suppose that I has a solution S. Then we schedule all the jobs corresponding to the items in S at time 0, each on a separate machine. Since S contains t items, and the number of machines is t as well, this is feasible. Moreover, the total resource requirement from the first resource is precisely A, whereas that from the second one is $\sum_{j\in S} a_{2,j} = \sum_{j\in S}(A - a_j) = (t-1)A$. The rest of the jobs are scheduled at time 1. Since their number is t, and since

$u_2 = 1$ is the second and last supply date, all the resources are supplied and the jobs can start promptly at time 1.

Conversely, suppose there is a feasible schedule of makespan two. Then, there are t jobs scheduled at time 0, and the remaining t jobs at time 1. Let S denote the set of the jobs scheduled at time 0. The resource requirements of those jobs in S equal the supply at time $u_1 = 0$, because $\sum_{j\in S} a_j = A$ follows from the resource constraints: on the one hand $\sum_{j\in S} a_j = \sum_{j\in S} a_{1,j} \le A$, and on the other hand $\sum_{j\in S} a_{2,j} = \sum_{j\in S}(A - a_j) = tA - \sum_{j\in S} a_j \le (t-1)A$, thus $A \le \sum_{j\in S} a_j$. Hence S is a feasible solution of the EVEN-PARTITION problem instance. $\square$

4. Arbitrary number of supplies and arbitrary release dates

Proof of Proposition 1. The main idea of the proof is that for any instance I of $P[m]|rm, r_j|F_{\max}$, and for any $\varepsilon > 0$, we construct an instance I' of the restricted problem, and show that after applying the ε-approximation algorithm A_ε to I', the resulting schedule S is feasible for I and satisfies the following condition:

$$F^S_{\max} \le (1+\varepsilon)F^*_{\max}(I') \le (1+\varepsilon)(F^*_{\max}(I) + \varepsilon u_q) \le (1+3\varepsilon)F^*_{\max}(I).$$

A_ε applied to I' implies the first inequality. The second one is the crux of the derivation and will be shown below, the third follows from $u_q \le F^*_{\max}(I)$. By Observation 1, the above derivation implies that we get a PTAS for $P[m]|rm, r_j|F_{\max}$.

Suppose that there are q supplies in instance I of $P[m]|rm|F_{\max}$: $u_1, u_2, \ldots, u_q$ with quantities $\tilde{b}_1, \tilde{b}_2, \ldots \tilde{b}_q$. We construct instance I' of the restricted problem: the $q' := \lceil 1/\varepsilon \rceil + 1$ (a constant for any fixed ε) supply dates are $u'_1 = 0$, $u'_\ell = (\ell - 1)\varepsilon u_q$ for $\ell = 2, \ldots, q' - 1$, and $u'_{q'} = u_q$. The amount of resource(s) supplied at u'_1 is $\tilde{b}'_1 := \tilde{b}_1$, and for u'_ℓ with $\ell \ge 2$ it is $\tilde{b}'_\ell = \sum_{\nu: u_\nu \le u'_\ell} \tilde{b}_\nu - \sum_{k<\ell} \tilde{b}'_k$ (see Figure 1). Notice that for each u_ℓ there is an $u'_{\ell'}$ with $u_\ell \le u'_{\ell'} < u_\ell + \varepsilon u_q$. Further on, the release date of each job is increased to the nearest u'_ℓ. Analogously to the supply dates, for each job release date r_j before u_q, there exists an u'_ℓ such that $r_j \le u'_\ell < r_j + \varepsilon u_q$. Besides, the two instances are the same.

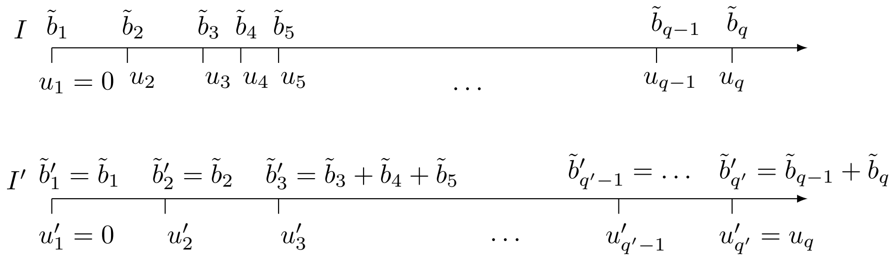

Figure 1: Supplies in case of an instance with an arbitrary number of supplies (above) and the corresponding instance with constant number of supplies (below).

Let S_I^* be an *optimal* schedule for I. If we increase the starting time of each job by εu_q, then the resulting schedule is a feasible solution of instance I', since the supplies, and the job release dates are delayed by less than εu_q. Hence, by using the properties of $F_{\max}$, $F^*_{\max}(I') \leq F^*_{\max}(I) + \varepsilon u_q$ follows. $\square$

5. PTAS for $Pm|rm = const, r_j|C_{\max}$

In this section first we provide a mathematical programming formulation of the problem, and then we prove Theorem 2.

5.1. A mathematical program for $P|rm, r_j|C_{\max}$

We can model $P|rm|C_{\max}$ with a mathematical program with integer variables. Let $\mathcal{M}$ denote the set of the machines and let $\mathcal{T}$ be the union of the set of supply dates and job release dates, i.e., $\mathcal{T} := \{u_\ell \mid \ell = 1, \dots, q\} \cup \{r_j \mid j \in \mathcal{J}\}$. Suppose $\mathcal{T}$ has τ elements, denoted by v_1 through v_τ, with $v_1 = 0$. We define the values $b_{\ell i} := \sum_{\nu \,:\, u_\nu \leq v_\ell} \tilde{b}_{\nu i}$ 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 $\tau \cdot |\mathcal{J}||\mathcal{M}|$ binary decision variables $x_{j\ell k}$, ($j \in \mathcal{J}, \ell = 1, \dots, \tau, k \in \mathcal{M}$) such that $x_{j\ell k} = 1$ if and only if job j is assigned to machine k and 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

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

s.t.

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

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

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

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

The objective function expresses the completion time of the job finished last using the observation that for every machine 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 machine and 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 machine k, and time point v_ℓ.

5.2. The PTAS

Let $p_{\text{sum}} := \sum_{j\in\mathcal{J}} p_j$ and note that $p_{\text{sum}} \le mC^*_{\max}$. Let $\varepsilon > 0$ be fixed. We can simplify the problem by applying Proposition 1, thus it is enough to deal with the case where $q = \lceil 1/\varepsilon \rceil + 1$, and $u_\ell = (\ell-1)\varepsilon u_q$ for $1 \le \ell < q$. Let $\mathcal{B} := \{j \in \mathcal{J} \mid p_j \ge \varepsilon^2 p_{\text{sum}}\}$ be the set of big jobs, and $\mathcal{S} := \mathcal{J} \setminus \mathcal{B}$ be 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, and $T^a := \mathcal{T} \setminus T^b$. Note that $|\mathcal{T}^b| = \lceil 1/\varepsilon \rceil$.

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 ($\sum_{k\in\mathcal{M}} \tilde{x}_{j\ell k} = 1$), satisfying either $v_\ell < u_q$, or $v_\ell = \max\{u_q, r_j\}$.*

The above statement is a generalization of the single machine case treated in Györgyi & Kis (2015b), and its proof can be found in Appendix A.

An *assignment of big jobs* is given by a partial solution $\hat{x}^{big} \in \{0,1\}^{\mathcal{B}\times\mathcal{T}\times\mathcal{M}}$ which assigns each big job to some machine k and time point v_ℓ. An assignment $\hat{x}^{big}$ of big jobs is *feasible* if the vector $\tilde{x} = (\hat{x}^{big}, 0) \in \{0,1\}^{\mathcal{J}\times\mathcal{T}\times\mathcal{M}}$ satisfies (2), (4) and also (3) for the big jobs. For a fixed feasible assignment $\hat{x}^{big}$ of big jobs, the supply from any resource i 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 $\bar{b}_{\ell i} := b_{\ell i} - \sum_{k\in\mathcal{M}}\sum_{j\in\mathcal{B}} a_{ij}\left(\sum_{\nu=1}^{\ell} x^{big}_{j\nu k}\right)$. Further on, let $\bar{C}^{\mathcal{B}}_\ell(k) := \max_{\omega=1,\ldots,\ell}(v_\omega + \sum_{\nu=\omega}^{\ell}\sum_{j\in\mathcal{B}} p_j x^{big}_{j\nu k})$ denote the earliest time point when the big jobs assigned to v_1 through v_ℓ may finish on machine k. Notice that $\bar{C}^{\mathcal{B}}_\ell(k) \geq v_\ell$ even if no big job is assigned to v_ℓ, or to any time period before v_ℓ.

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 $v_\ell \in \mathcal{T}^b$, $k \in \mathcal{M}$:*

$$\sum_{j\in\mathcal{S}^b} p_j \hat{x}^{small}_{j\nu k} \leq \max\{0, v_{\ell+1} - \bar{C}^{\mathcal{B}}_\ell(k)\} + \varepsilon^2 p_{\text{sum}}. \tag{6}$$

The above statement is an easy generalization of the single machine case treated in Györgyi & Kis (2015b), see the proof there.

For every feasible big job assignment we will determine a complete solution of (1)-(5). We search these solution in two steps: first we assign the small jobs to time moments and then to machines. Let $x_{j\ell} := \sum_{k\in\mathcal{M}} x_{j\ell k}$. Now, the linear program is defined with respect to any feasible assignment $\hat{x}^{big}$ of the big jobs:

$$\max \sum_{v_\ell\in\mathcal{T}^b}\sum_{j\in\mathcal{S}^b} p_j x^{small}_{j\ell} \tag{7}$$

s.t.

$$\sum_{j\in\mathcal{S}^b}\sum_{\nu=1}^{\ell} a_{ij}x^{small}_{j\nu} \le \bar{b}_{\ell i}, \quad v_\ell\in\mathcal{T}^b,\ i\in\mathcal{R} \tag{8}$$

$$\sum_{j\in\mathcal{S}^b} p_j x^{small}_{j\ell} \le \sum_{k=1}^{m}\max\{0, v_{\ell+1}-\bar{C}^{\mathcal{B}}_{\ell}(k)\} + m\varepsilon^2 p_{\mathrm{sum}}, \quad v_\ell\in\mathcal{T}^b \tag{9}$$

$$\sum_{v_\ell\in\mathcal{T}^b\cup\{u_q\}} x^{small}_{j\ell} = 1, \quad j\in\mathcal{S}^b \tag{10}$$

$$x^{small}_{j\ell} = 0, \quad j\in\mathcal{S}^b,\ v_\ell\in\mathcal{T} \text{ such that } v_\ell<r_j, \text{ or } v_\ell>u_q \tag{11}$$

$$x^{small}_{j\ell} \ge 0, \quad j\in\mathcal{S}^b,\ v_\ell\in\mathcal{T}. \tag{12}$$

The objective function (7) maximizes the total processing time of those small jobs assigned to some time point v_ℓ *before* u_q. Constraints (8) make sure that no resource is overused taking into account the fixed assignment of big jobs as well. Inequalities (9) ensure that the total processing time of those small jobs assigned to $v_\ell \in \mathcal{T}^b$ does not exceed the total size of all the gaps on the m machines between v_ℓ and $v_{\ell+1}$ by more than $m\varepsilon^2 p_{\mathrm{sum}}$. Due to (10), 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 (11). Finally, we require that the values $x^{small}_{j\ell}$ 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 $\bar{x}^{small}_{j\ell}=1$, otherwise it is *fractional*. Throughout the algorithm we maintain the best schedule found so far, S^{best}, and its makespan $C_{\max}(S^{best})$.

The following notion is repeatedly used in the algorithms of this paper. Suppose we have a partial schedule $\tilde{S}$ and consider an idle period I on some machine M_k. Suppose j_1 is not scheduled in $\tilde{S}$, and we schedule j_1 on M_k with starting time $t_1 \in I$. This transforms $\tilde{S}$ as follows. For each job j scheduled on M_k in $\tilde{S}$ with $\tilde{S}_j > t_1$, let $P_k[t_1, \tilde{S}_j]$ denote the total processing time of those jobs scheduled on M_k in $\tilde{S}$ between t_1 and $\tilde{S}_j$. We update the start-time of j to $\max\{\tilde{S}_j, t_1 + p_{j_1} + P_k[t_1, \tilde{S}_j]\}$. The start time of all other jobs do not change.

After all these preliminaries, the PTAS is as follows.

Algorithm A

Initialization: S^{best} is a schedule where each job is scheduled on M_1 after $\max\{r_{\max}, u_q\}$.

1. Assign the big jobs to time points v_1 through v_τ and to machines 1 through $|\mathcal{M}|$ in all possible ways which satisfy Proposition 2, and for each feasible assignment x^{big} do steps 2 - 7 :
2. Define and solve linear program (7)-(12), 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. Invoke Subroutine Sch with $\bar{\mathcal{J}} := \mathcal{B}$ to create a partial schedule S^{part} from the big jobs.
5. The next procedure schedules all the small jobs assigned to a time point before u_q. For each $v_\ell \in \mathcal{T}^b$ do:
   i) Put the small jobs with $\bar{x}^{small}_{j\ell} = 1$ into a list in an arbitrary order.
   ii) For $k = 1, \dots, m$ do the following steps:
      a) Let t be such that the total processing time of the first t jobs from the ordered list is in $[\max\{0, v_{\ell+1} - \bar{C}^{\mathcal{B}}_\ell(k)\} + \varepsilon^2 p_{\text{sum}}, \max\{0, v_{\ell+1} - \bar{C}^{\mathcal{B}}_\ell(k)\} + 2\varepsilon^2 p_{\text{sum}}]$. If no such t exists (since there are not enough jobs left), then let t be the current number of the small jobs in the ordered list.
      b) Assign the first t jobs from the list to machine k, and schedule all of them (as a single job) starting from the earliest idle time on M_k after $\bar{C}^{\mathcal{B}}_\ell(k)$. Finally, delete them from the ordered list.

   Let $C^{part}_{\max}$ denote the makespan of S^{part} after this step.

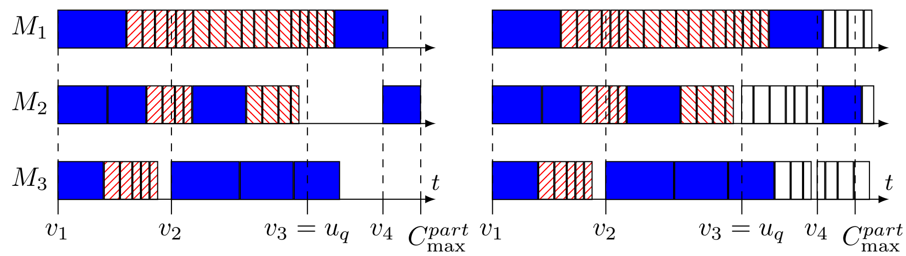

Figure 2: A partial schedule after Step 5 on the left (big jobs are blue, small jobs are hatched) and a complete schedule on the right. The jobs scheduled at Step 6 are white. Each job scheduled after v_4 has a release date v_4, since M_3 is idle before v_4.

6. Schedule the remaining small jobs one by one in non-decreasing release date order $(J_1, J_2, \ldots)$. Let J_j be the next job to be scheduled, and M_k a machine with the earliest idle time after $\max\{u_q, r_j\}$ in the current schedule. Schedule J_j on this machine at that time, and let $x^{small}_{j\ell k} = 1$, where $\max\{u_q, r_j\} = v_\ell \in \mathcal{T}$. Let S^{act} be the resulting schedule.

7. If $C_{\max}(S^{act}) < C_{\max}(S^{best})$, then let $S^{best} := S^{act}$.

8. After examining each feasible assignment of the big jobs, output S^{best}.

Subroutine Sch

Input: $\bar{\mathcal{J}} \subseteq \mathcal{J}$ and $\bar{x}$ such that for each $j \in \bar{\mathcal{J}}$ there exists a unique (ℓ, k) with $\bar{x}_{j\ell k} = 1$.

Output: partial schedule S^{part} of the jobs in $\bar{\mathcal{J}}$.

1. S^{part} is initially empty, then we schedule the jobs on each machine in increasing v_ℓ order (first we schedule those jobs assigned to v_1, and then those assigned to v_2, etc.):

2. When scheduling the next job with $\bar{x}_{j\ell k} = 1$, then it is scheduled at time $\max\{v_\ell, C_{last}(k)\}$, where $C_{last}(k)$ is the completion time of the last job scheduled on machine M_k, or 0 if no job has been scheduled yet on M_k.

See Figure 2 for illustration. We will prove that the solution found by Algorithm A is feasible for (1)-(5), its value is not far from the optimum, and the algorithm runs in polynomial time.

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

Proof. At the end of the algorithm each job is scheduled exactly once sometime after its release date, thus the solution satisfies (3), (4) and (5). The algorithm examines only feasible assignments of the big jobs, hence these jobs cannot violate the resource constraints. Since $\bar{x}^{small}$ is a feasible solution of (7) - (12) and $\sum_{k\in\mathcal{M}} x_{j\ell k} = x_{j\ell}$, $(\forall j \in \mathcal{J})$, thus the assignment corresponds to S^{part} satisfies (2). Finally, since u_q is the last time point when some resource is supplied, thus when the algorithm schedules the remaining jobs at Step 6, the constraints (2) remain feasible. $\square$

To prove that the makespan of the schedule found by the algorithm is near to the optimum, we need Propositions 4 and 5. From these we conclude that the fractionally assigned jobs and the 'errors' in (9) do not cause big delays. We utilize that the number of the release dates before u_q is a constant. From Proposition 5 we can deduce that, in case of appropriate big job assignment, $C_{\max}^{part}$ is not much bigger than $C_{\max}^*$. If the makespan of the constructed schedule is larger than $C_{\max}^{part}$, then the machines finish the jobs nearly at the same time, thus we can prove that there are no big delays relative to an optimal schedule.

Proposition 4. *In any basic solution of the linear program (7)-(12), 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 assign 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 $\gamma = |\mathcal{S}^b|+(|\mathcal{R}|+1)\cdot|\mathcal{T}^b|$ constraints. Therefore, in $\bar{x}^{small}$ there are at most γ 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. $\square$

Proposition 5. *Consider a big job assignment after Step 1. Let S^{big} denote the partial schedule of this assignment and $C^{\mathcal{B}}_{\max}$ its makespan.*

1. *If a big job J_j is assigned to v_ℓ at Step 1, then $S^{part}_j \leq S^{big}_j + 2\varepsilon^2(\ell-1)p_{\text{sum}}$.*
2. *$C^{part}_{\max} \leq \max\{u_q, C^{\mathcal{B}}_{\max}\} + 2\varepsilon^2|\mathcal{T}^b|p_{\text{sum}}$.*

Proof. Recall that the jobs assigned to the same time point and machine are in non-increasing processing time order.

1. The algorithm can push to the right the start time of big job assigned to some v_ℓ at Step 5(ii)a, or in other words, when it schedules some small jobs before v_ℓ. However, this can happen only $\ell - 1$ times, thus the claim follows.
2. Imagine a fictive big job starts at $\max\{u_q, C^{\mathcal{B}}_{\max}\}$, and apply the first part of the proposition.

$\square$

Lemma 2. *The algorithm constructs at least one feasible schedule of makespan at most $(1 + O(|\mathcal{T}^b|\varepsilon^2))$ times the optimum makespan $C^*_{\max}$.*

Proof. By Lemma 1, the algorithm outputs a feasible schedule. Consider an optimal schedule S^* and the corresponding solution $(\hat{x}^{big}, \hat{x}^{small})$ of (1)-(5) that satisfies Proposition 3. The algorithm will examine $\hat{x}^{big}$, since it is a feasible big job assignment. Let $C_{\max}$ denote the makespan of the schedule S found by the algorithm in this case. The observation below follows from Proposition 5:

Observation 2. $C^{part}_{\max} \leq C^*_{\max} + 2|\mathcal{T}^b|\varepsilon^2 p_{\text{sum}}$.

If no small job scheduled at Step 6 starts after $C^{part}_{\max} - \varepsilon^2 p_{\text{sum}}$, then the statement of the lemma follows from Observation 2 since $p_{\text{sum}} \leq mC^*_{\max}$ and $C_{\max} \leq C^{part}_{\max} + \varepsilon^2 p_{\text{sum}}$, thus $C_{\max} \leq (1 + (2|\mathcal{T}^b| + 1)m\varepsilon^2)C^*_{\max}$.

From now on, suppose that at least one small job scheduled at Step 6 starts after $C_{\max}^{part} - \varepsilon^2 p_{\text{sum}}$. For similar reasons, also suppose that $C_{\max} > \max\{C_{\max}^{part}, v_\tau\} + \varepsilon^2 p_{\text{sum}}$ (this means that for every machine there is at least one small job that starts after $\max\{C_{\max}^{part}, v_\tau\}$ and scheduled at Step 6).

Observation 3. *The difference between the finishing time of two arbitrary machines is at most $\varepsilon^2 p_{\text{sum}}$.*

We prove the statement of the lemma with Claims 1, 2 and 3.

Claim 1. *If there is no gap on any machine, then $C_{\max} \leq (1 + m\varepsilon^2)C_{\max}^*$.*

Proof. According to Observation 3 each machine is working between 0 and $(C_{\max} - \varepsilon^2 p_{\text{sum}})$. Therefore $C_{\max}^* \geq C_{\max} - \varepsilon^2 p_{\text{sum}}$ which implies $C_{\max} \leq (1 + m\varepsilon^2)C_{\max}^*$. □

Claim 2. *If the last gap finishes after u_q, then $C_{\max} \leq (1+(2|\mathcal{T}^b|+1)m\varepsilon^2)C_{\max}^*$.*

Proof. Note that this gap must finish at a release date r_{j_0}. Notice that each small job scheduled after r_{j_0} has a release date at least r_{j_0} or else we would have scheduled that job into the last gap, thus

Observation 4. *The small jobs starting after r_{j_0} in S are scheduled after r_{j_0} in S^*.*

Consider an arbitrary machine M_k and the last big job J_j that is starting before r_{j_0} on this machine in S^*. If $S_j^{part} < u_q$ or there is no gap between u_q and S_j^{part} in S^{part}, then we have not scheduled any job on M_k before J_j at Step 6, thus the starting (and the completion) time of J_j is at most $2|\mathcal{T}^b|\varepsilon^2 p_{\text{sum}}$ later in S than in S^* (Proposition 5). Otherwise the starting time of J_j is the same in S^{part} and in S^* ($S_j^{part} = S_j^*$), since we can suppose that the jobs assigned to the same time point and machine are scheduled in the same non-increasing processing time order. If we push S_j at Step 6 once, then we cannot schedule any more jobs before S_j in a later step, thus we can push S_j by at most $\varepsilon^2 p_{\text{sum}}$ in total, thus

Observation 5. *If $J_j \in \mathcal{B}$, then $S_j \leq S_j^* + 2|\mathcal{T}^b|\varepsilon^2 p_{\text{sum}}$.*

Suppose that a job J_j is scheduled from S'_j to $C'_j = S'_j + p_j$ in a schedule S' and $S'_j \leq t \leq C'_j$. In this case we can divide J_j into two parts: to the part of J_j that is scheduled before t (it has a processing time of $t - S'_j$) and to the part that is scheduled after t (it has a processing time of $C'_j - t$). Suppose that t is fixed and we divided all the jobs such that $S'_j \leq t \leq C'_j$ into two parts. Let $P_b^{(t)}(S')$ denote the total processing time of the jobs and job parts that are scheduled before t in S' and $P_a^{(t)}(S')$ denote the same after t ($P_b^{(t)}(S') + P_a^{(t)}(S') = p_{\text{sum}}$).

Observation 6. $P_a^{(r_{j_0}+2|\mathcal{T}^b|\varepsilon^2 p_{\text{sum}})}(S) \leq P_a^{(r_{j_0})}(S^*)$ *(follows from Observations 4 and 5).*

Let $P := P_a^{(r_{j_0}+2|\mathcal{T}^b|\varepsilon^2 p_{\text{sum}})}(S)$. Since there is no gap after r_{j_0} in S, $C_{\max} \leq r_{j_0} + 2|\mathcal{T}^b|\varepsilon^2 p_{\text{sum}} + (P/m + \varepsilon^2 p_{\text{sum}})$ follows from Observation 3. Since $C^*_{\max} \geq r_{j_0} + P/m$ (from Observation 6), thus $C_{\max} \leq C^*_{\max} + (2|\mathcal{T}^b| + 1)\varepsilon^2 p_{\text{sum}} \leq (1 + (2|\mathcal{T}^b| + 1)m\varepsilon^2)C^*_{\max}$, therefore we have proved Claim 2. $\square$

For a schedule S', let $S'_{\mathcal{B}}$ denote the schedule of the big jobs (where the big jobs have the same starting times as in S' and the small jobs are deleted from S') and $S'_{\mathcal{S}}$ denote the schedule of the small jobs (similarly).

Claim 3. *If each gap finishes before u_q, then $C_{\max} \leq (1+((2|\mathcal{T}^b|+1)m+(|\mathcal{R}|+1) \cdot |\mathcal{T}^b|)\varepsilon^2)C^*_{\max}$.*

Proof. See Appendix A. $\square$

The lemma follows from Claims 1, 2 and 3. $\square$

Lemma 3. *For any fixed $\varepsilon > 0$, the running time of the algorithm is polynomial in the size of the input if $|\mathcal{T}^b|$ is a constant.*

Proof. Since the processing time of each big job is at least $\varepsilon^2 p_{\text{sum}}$, the number of the big jobs is at most $\lfloor 1/\varepsilon^2 \rfloor$, a constant, since ε is a constant by assumption. Thus, the total number of assignments of big jobs to time point in $\mathcal{T}^b$ and to machine in $\mathcal{M}$ is also constant $O((m/\varepsilon)^{1/\varepsilon^2})$. For each feasible assignment, a linear program of polynomial size in the input and in $1/\varepsilon$ must be solved. This can be accomplished by the Ellipsoid method in polynomial time, see Gács & Lovász (1981). The remaining steps (rounding the solution, machine assignment and scheduling the small jobs) are obviously polynomial ($O(n \log n)$). $\square$

Proof of Theorem 2. Since $q = \lceil 1/\varepsilon \rceil + 1$, we get that $|\mathcal{T}^b| = q - 1$ in the transformed instances. Therefore, by Lemma 2, the performance ratio of the algorithm is $(1+O(|\mathcal{T}^b|\varepsilon^2)) = (1+O(\varepsilon))$, where the constant factor c in $O(\cdot)$ does not depend on the input or on $1/\varepsilon$. However, by Observation 1 this is sufficient to have a PTAS. Finally, the polynomial time complexity of the algorithm in the size of the input was shown in Lemma 3. $\square$

Remark 1. *Note that if a job is assigned to a v_ℓ, then $S_j \geq v_\ell$ at the end of the algorithm and each schedule such that this is true cannot violate the resource constraint. Suppose that we fixed a big job assignment and solved the LP. Then*

- *if $j \in \mathcal{S}^a$, then let $\bar{r}_j := r_j$.*
- *if $j \in \mathcal{S}^b \cup \mathcal{B}$ and $\exists \ell : x_{j\ell} = 1$, then let $\bar{r}_j := v_\ell$.*
- *otherwise, let $\bar{r}_j := u_q$.*

After that, use the PTAS of Hall & Shmoys (1989) for the problem $P|\bar{r}_j|C_{\max}$. It is easy to prove that the schedule obtained is feasible and its makespan is at most $(1+\varepsilon)$ times the makespan of the schedule created by Algorithm A, thus it is also a PTAS for our problem. The algorithm of Hall and Shmoys works for an arbitrary number of machines, however this number must be a constant when applied to our problem, otherwise the error bound breaks down.

6. $Pm|rm = const, r_j, ddc|C_{\max}$

Suppose that there is a dedicated machine for each job, or in other words, the assignment of jobs to machines is given in the input. Let M_{k_j} denote the machine on which we have to schedule J_j and $\mathcal{J}_k$ denote the set of jobs dedicated to M_k. We can model this problem with the IP (1)-(5) if we drop all the variables $x_{j\ell k}$ where $k \neq k_j$. Let us denote this new IP by (1')-(5'). We prove that there is a PTAS for this problem. The main idea of the algorithm is the same as in the previous section, however there are important differences, since we cannot balance the finishing time of the machines with the small jobs after u_q (cf. Observation 3).

Let $\varepsilon > 0$ be fixed. According to Proposition 1, we can assume that q and the number of distinct job release dates until u_q are at most $\lceil 1/\varepsilon \rceil + 1$. Divide the set of jobs into big and small ones ($\mathcal{B}$ and $\mathcal{S}$), and schedule them separately. These sets are the same as in Section 5. We assign the big jobs to time points in all possible ways (cf. Proposition 2). Notice that since $|\mathcal{B}| \leq 1/\varepsilon^2$, which is a constant because $\varepsilon > 0$ is fixed, the number of big job assignments is polynomial in the size of the input. We perform the remaining part of the algorithm for each big job assignment. The first difference from the previous PTAS is the following: now we assign each small job in $\mathcal{S}^a$ to its release date and then we create the schedule S^1 from this partial assignment. Let $C^1_{\max}$ denote the makespan of S^1 and I_k the total idle time on machine k between u_q and $C^1_{\max}$ (if $C^1_{\max} \leq u_q$, then $I_k = 0$ for all $k \in \mathcal{M}$).

We have to schedule the small jobs in $\mathcal{S}^b$. We will schedule them in a suboptimal way and finally we choose the schedule with the lowest makespan. We will prove that the best solution found by the algorithm has a makespan of no more than $(1+\varepsilon)C^*_{\max}$ and the algorithm has a polynomial complexity.

For a fixed partial schedule we define the following linear program:

$$\min \bar{P} \tag{13}$$

s.t.

$$\sum_{j \in \mathcal{S}^b, v_\ell \geq u_q, k_j = k} p_j x^{small}_{j\ell k_j} \leq I_k + \bar{P}, \quad k \in \mathcal{M} \tag{14}$$

$$\sum_{j \in \mathcal{S}^b} \sum_{\nu=1}^{\ell} a_{ij} x^{small}_{j\nu k_j} \leq \bar{b}_{\ell i}, \quad v_\ell \in \mathcal{T}^b,\ i \in \mathcal{R} \tag{15}$$

$$\sum_{j \in \mathcal{S}^b, k_j = k} p_j x^{small}_{j\ell k_j} \leq \max\{0, v_{\ell+1} - \bar{C}^{\prime\mathcal{B}}_{\ell}(k)\} + \varepsilon^2 p_{\text{sum}}, \quad v_\ell \in \mathcal{T}^b,\ k \in \mathcal{M} \tag{16}$$

$$\sum_{v_\ell \in \mathcal{T}} x^{small}_{j\ell k_j} = 1, \quad j \in \mathcal{S}^b \tag{17}$$

$$x^{small}_{j\ell k_j} = 0, \quad j \in \mathcal{S}^b,\ v_\ell \in \mathcal{T} \text{ such that } v_\ell < r_j, \text{ or } v_\ell > u_q \tag{18}$$

$$\bar{P} \geq 0 \tag{19}$$

$$x^{small}_{j\ell k_j} \geq 0, \quad j \in \mathcal{S}^b,\ v_\ell \in \mathcal{T}. \tag{20}$$

The notations are the same as before. Our objective ($\bar{P}$) is to minimize the increase of the makespan compared to $C^1_{\max}$. The PTAS is as follows:

Algorithm B

Initialization: S^{best} is a schedule where each job is scheduled after $\max\{r_{\max}, u_q\}$ (in an arbitrary order without any idle time) on its dedicated machine.

1. Assign the big jobs to time points v_1 through v_τ which satisfies Proposition 2, and for each feasible assignment x^{big} do steps 2 - 7 :
2. Assign each small jobs in $\mathcal{S}^a$ to its release date, i.e., $x^a_{j\ell k_j} = 1$ if and only if $j \in \mathcal{S}^a$ and $r_j = v_\ell \in \mathcal{T}^a$. Invoke Subroutine Sch with $\bar{\mathcal{J}} = \mathcal{B} \cup \mathcal{S}^a$ and $\bar{x} = (x^{big}, x^a, 0)$. Let $C^1_{\max} := C_{\max}(S^{part})$.
3. Define and solve linear program (13)-(20), and let $\bar{x}^{small}$ be an optimal basic solution.
4. 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 $\bar{x}^{small}$.
5. Using Subroutine Sch, create a new partial schedule S^{part} for the subset of jobs $\bar{\mathcal{J}} = \mathcal{B} \cup \mathcal{S}^a \cup (\mathcal{S}^b \setminus U)$, and assignment $\bar{x} = (x^{big}, x^a, x^{small})$. Let $C^{part}_{\max}$ denote the makespan of this schedule (S^1 is not used). The next step inserts the remaining jobs into S^{part}.
6. Schedule the remaining small jobs one by one in non-decreasing release date order $(J_1, J_2, \ldots)$. Let J_j be the next job to be scheduled. Schedule J_j on

M_{k_j} at the earliest idle time after $\max\{u_q, r_j\}$ in the current schedule and let $x^{small}_{j\ell k_j} = 1$, where $\max\{u_q, r_j\} = v_\ell \in \mathcal{T}$. Let S^{act} be the resulting schedule.

7. If the makespan of the resulting schedule (S^{act}) is smaller than $C_{\max}(S^{best})$, then let $S^{best} := S^{act}$.

8. After examining each feasible assignment of the big jobs, output S^{best}.

Lemma 4. *Every complete solution (x^{big}, x^{small}) constructed by the algorithm is feasible for (1')-(5').*

Proof. (2') follows from (15) (the jobs scheduled after u_q cannot violate this constraint), while the other constraints are obviously met. □

Proposition 6. *In any basic solution of the linear program (7)-(12), there are at most $(|\mathcal{R}| + 1) \cdot |\mathcal{T}^b|$ fractional jobs.*

Proof. Similar to Proposition 4. □

Proposition 7. *1. If a job J_j is assigned to v_ℓ at Step 1 or 2, then $S^{part}_j \le S^1_j + \min\{\ell - 1, |\mathcal{T}^b|\}\varepsilon^2 p_{\text{sum}}$.*

2. $C^{part}_{\max} \le \max\{u_q, C^1_{\max}\} + \bar{P} + |\mathcal{T}^b|\varepsilon^2 p_{\text{sum}}$.

Proof. Similar to Proposition 5. □

Lemma 5. *The algorithm constructs at least one feasible schedule of makespan at most $(1 + O(|\mathcal{T}^b|\varepsilon^2))$ times the optimum makespan $C^*_{\max}$.*

Proof. By Lemma 4, the algorithm outputs a feasible schedule. Consider an optimal schedule S^* and the corresponding solution $(\hat{x}^{big}, \hat{x}^{small})$ of (1')-(5') that satisfies Proposition 2. The algorithm will examine $\hat{x}^{big}$, since it is a feasible big job assignment. The partial assignment of the small jobs in $\mathcal{S}^b$ in S^* determines a feasible solution of (13)-(20), thus $\max\{u_q, C^1_{\max}\} + \bar{P} \le C^*_{\max}$.

According to Proposition 7 $C^{part}_{\max} \le \max\{u_q, C^1_{\max}\} + \bar{P} + |\mathcal{T}^b|\varepsilon^2 p_{\text{sum}}$, and $C_{\max} \le C^{part}_{\max} + (|\mathcal{R}| + 1) \cdot |\mathcal{T}^b|\varepsilon^2 p_{\text{sum}}$ follows from Proposition 6. Therefore $C_{\max} \le (1 + ((|\mathcal{R}| + 2) \cdot |\mathcal{T}^b|) m\varepsilon^2) C^*_{\max}$. □

Lemma 6. *For any fixed $\varepsilon > 0$, the running time of the algorithm is polynomial in the size of the input.*

Proof. Similar to Lemma 3. □

Proof of Theorem 3. Since $|\mathcal{T}^b| = q - 1$ (Proposition 1), the theorem follows from Lemmas 5 and 6. □

Remark 2. *Suppose that, there is a dedicated machine for each job in a given set $\mathcal{J}' \subset \mathcal{J}$ and we can schedule each job in $\mathcal{J} \setminus \mathcal{J}'$ on any machine. We still have a PTAS for this case: the main difference is that at Step 6 we first have to schedule the jobs in $\mathcal{J}'$ and then the remaining jobs similarly to Step 6 in Algorithm A.*

7. $Pm|rm = 1, p_j = a_j|L_{\max}$

In this section we prove Theorem 4. Throughout this section we assume that $\varepsilon > 0$ is a small constant with $1/\varepsilon \in \mathbb{Z}$. Let $\mathcal{S}' := \{j \in \mathcal{J} | p_j \leq \varepsilon^2 u_q\}$ be the set of *tiny* jobs, and $\mathcal{B}' := \mathcal{J} \setminus \mathcal{S}'$ be the set of *huge* jobs. Note that this partition is quite different from the one in Section 5. According to Proposition 1, we can assume that $q = 1/\varepsilon + 1$, and $u_\ell = (\ell - 1)\varepsilon u_q$ ($\ell = 1, 2, \ldots, q - 1$). Note that between two consecutive supply dates at most $1/\varepsilon$ huge jobs can start, thus we can assume $\sum_{j \in \mathcal{B}'} x_{j\ell k} \leq 1/\varepsilon$, if $\ell < q$ and $k \in \mathcal{M}$, therefore there are at most $(n + 1)^{(1/\varepsilon)qm}$ different assignments of huge jobs to the supply dates u_1 through u_{q-1}. We can examine all of them, since m and ε are constants. The remaining huge jobs are assigned to u_q, but we assign them to machines later. For each huge job assignment we will guess approximately the total processing time of those tiny jobs that start in the interval $[u_\ell, u_{\ell+1})$ on machine M_k, $\ell = 1, \ldots, q - 1$, and $k = 1, \ldots, m$. A guess is a number of the form $g_{k,\ell} \cdot (\varepsilon^2 u_q)$, where $0 \leq g_{k,\ell} \leq 1/\varepsilon + 1$ is an integer. A guess for all the $q - 1$ supply dates and all the m machines can be represented by a $m \times (q - 1)$-tuple $g = (g_{k,\ell})$, and let G denote the set of all possible guesses. The algorithm is as follows:

Algorithm C

Initialization: S^{best} is a schedule where each job is scheduled on M_1 after u_q.

1. For each feasible partial assignment $\hat{x}^{huge,b}$ of huge jobs to machines and supply dates u_1 through u_{q-1}, perform the following steps.
2. For each tuple $g \in G$, do steps 3 - 6:
3. We create a feasible partial assignment $\hat{x}^b$ by assigning also the tiny jobs to machines and supply dates u_1 through u_{q-1}. Initially $\hat{x}^b$ is the same as $\hat{x}^{huge,b}$. Let L be the list of tiny jobs sorted in non-decreasing d'_j order. Jobs from L are assigned to machines and to supply dates u_1 through u_{q-1} until all jobs from L get assigned or all the supply dates from u_1 through u_{q-1} are processed. When processing supply date u_ℓ, $\ell \in \{1, \ldots, q-1\}$, we first assign jobs to M_1, then to M_2, etc. Let M_k be the next machine to receive some jobs. Let $h_{k,\ell}$ be the smallest number of tiny jobs from the beginning of L with a total processing time of at least $g_{k,\ell}(\varepsilon^2 u_q)$, and let $z_{k,\ell}$ be the maximum number of tiny jobs from the beginning of L that can be assigned to u_ℓ without violating the resource constraint. Assign $\min\{h_{k,\ell}, z_{k,\ell}\}$ jobs from the beginning of L to supply date u_ℓ on M_k, and remove them from L. Then proceed with the next machine until all machines are processed or L becomes empty.
4. Create a partial schedule S^{part} from $\hat{x}^b$ with the following modification of subroutine Sch (5): always schedule first the tiny jobs and then the huge jobs if they are assigned to the same machine M_k and to the same supply date u_ℓ.
5. Let $C^{part}_{\max}(k)$ be the time when M_k finishes S^{part}. Invoke the algorithm of Appendix B with $\max\{C^{part}_{\max}(k), u_q\}$ amount of preassigned work on M_k ($k = 1, 2, \ldots, m$) to schedule the remaining jobs. Let S^{act} be the resulting schedule.
6. If $L'_{\max}(S^{act}) < L'_{\max}(S^{best})$, then let $S^{best} := S^{act}$.
7. After examining each feasible assignment of huge jobs before u_q, output S^{best}.

The final schedule S^{best} is obviously feasible and the running time of the algorithm is polynomial in the size of the input, since the number of possible huge job assignments before u_q can be bounded by $O((n+1)^{(1/\varepsilon)qm})$, the number of the tuples is $(1/\varepsilon+2)^{m(q-1)}$, steps 3 and 4 require $O(n \log n)$ time, while step 5 also requires polynomial time (Hall & Shmoys (1989), Appendix B).

For the sake of proving that Algorithm C is a PTAS, we construct an intermediate schedule $\tilde{S}$ which, on the one hand, has a similar structure to that of an optimal schedule, and on the other hand, not far from the schedule computed by Algorithm C. $\tilde{S}$ is derived from an optimal schedule S^* as follows. Let $g^*_{k,\ell}$ ($k \in \{1, \dots, m\}$ and $\ell \in \{1, \dots, q-1\}$) be the smallest integer such that $(g^*_{k,\ell} - 1) \cdot (\varepsilon^2 u_q)$ is at least the total processing time of the tiny jobs starting in $[u_\ell, u_{\ell+1})$ on M_k in S^* unless there is no such tiny job, in which case $g^*_{k,\ell} = 0$. First perform Steps 3 and 4 of Algorithm C with the partial huge job assignment $(x^{huge,b})^*$ that corresponds to S^*, and the tuple g^* just defined. After that, schedule the remaining huge jobs at $\tilde{S}_j := S^*_j + 5\varepsilon u_q$ on the same machine as in S^* and finally schedule the remaining tiny jobs in earliest-due-date (EDD) order after $\max\{C^{part}_{\max}, u_q\}$ at the earliest idle time on any machine.

In order to compare $\tilde{S}$ with S^{best} (Proposition 8), and with S^* (Proposition 9), first we make two observations. Let $\tilde{\mathcal{J}}_{\ell,k}$ denote the set of tiny jobs that are assigned to u_ℓ and M_k in $\tilde{S}$ and $\mathcal{J}^*_{\ell,k}$ denote the set of tiny jobs with $u_\ell \leq S^*_j < u_{\ell+1}$ on machine k. $\tilde{\mathcal{J}}_\ell := \cup_k \tilde{\mathcal{J}}_{\ell,k}$ and $\mathcal{J}^*_\ell := \cup_k \mathcal{J}^*_{\ell,k}$. Let $\mathcal{M}^*_\ell$ denote the set of those machines with at least one tiny job that starts in $[u_\ell, u_{\ell+1})$ in S^*.

Observation 7. *For each $\ell < q$ and $M_k \in \mathcal{M}$, $p(\tilde{\mathcal{J}}_{\ell,k}) < p(\mathcal{J}^*_{\ell,k}) + 3\varepsilon^2 u_q$ and $p(\cup_{\nu \leq \ell} \tilde{\mathcal{J}}_\nu) \geq p(\cup_{\nu \leq \ell} \mathcal{J}^*_\nu) - \varepsilon^2 u_q$.*

Proof. See Appendix A. □

Observation 8. *After processing supply date u_ℓ in Step 3 of Algorithm C, then at least one of the following conditions holds: (i) there is not enough resource to assign the next tiny job, (ii) $p(\cup_{\nu \leq \ell} \tilde{\mathcal{J}}_\nu) \geq p(\cup_{\nu \leq \ell} \mathcal{J}^*_\nu)$ or (iii) $\mathcal{M}^*_\ell = \emptyset$.*

Proof. If (i) and (iii) are not true, then we have $p(\mathcal{J}^*_\ell) \leq \sum_{k \in \mathcal{M}^*_\ell} (g^*_{k,\ell} - 1) \cdot (\varepsilon^2 u_q) \leq p(\tilde{\mathcal{J}}_\ell) - \varepsilon^2 u_q$, where the first inequality follows from the definition of g^*, the second from the rule of Algorithm C (step 3). Consequently, the observation follows from the second part of Observation 7 (using it for $\ell - 1$). □

Proposition 8. *$\tilde{S}$ is feasible, and $L'_{max}(S^{best}) \leq (1 + \varepsilon) L'_{\max}(\tilde{S})$.*

Proof. $\tilde{S}$ cannot violate the resource constraints by the rules of Algorithm C, and due to Observation 7, the jobs scheduled on an arbitrary machine M_k must end before a huge job scheduled in the last stage of the construction of $\tilde{S}$ would start, since for all those huge jobs, $\tilde{S}_j = S_j^* + 5\varepsilon u_q$ by definition. In some iteration, Algorithm C will consider the huge job assignment and the tuple that we used to define $\tilde{S}$. Hence, after step 4, $\tilde{S}$ and S^{part} coincide. Therefore, the Proposition follows from Hall & Shmoys (1989) and Appendix B. □

Proposition 9. $L'_{\max}(\tilde{S}) \leq L'_{\max}(S^*) + 6\varepsilon u_q$.

Proof. Let j be such that $L'_j(\tilde{S}) = L'_{\max}(\tilde{S})$. First suppose that j is huge. If j is scheduled at step 4 (since it is assigned to a supply date u_ℓ and a machine M_k), then the jobs assigned to M_k and to a $u_{\ell'}$ with $\ell' < \ell$, are completed at most $3(\ell-1)\varepsilon^2 u_q$ later in $\tilde{S}$ than the jobs with $S^*_{j'} < u_\ell$ on M_k in S^* (Observation 7). The total processing time of the jobs that are assigned to u_ℓ and M_k and scheduled before j in $\tilde{S}$ is at most $\varepsilon u_q + 3\varepsilon^2 u_q$, thus $\tilde{C}_j \leq C^*_j + 5\varepsilon u_q$ follows. If it is scheduled at step 5, then originally we have $\tilde{S}_j = S^*_j + 5\varepsilon u_q$ and we may push j to the right by at most $\varepsilon^2 u_q$, thus $\tilde{C}_j \leq C^*_j + 6\varepsilon u_q$.

Now suppose that j is tiny.

Claim 4. $\min\{d_{j'} : j' \in \cup_{\nu \geq \ell} \tilde{\mathcal{J}}_\nu\} \geq \min\{d_{j'} : j' \in \cup_{\nu \geq \ell} \mathcal{J}^*_\nu\}$, *for each* $\ell \leq q$.

Proof. See Appendix A. □

If j is assigned to an u_ℓ with $\ell < q$, then according to Claim 4, there exists a job j^* with $d_{j^*} \leq d_j$ and $S^*_{j^*} \geq u_\ell$. Let M_k be the machine which processes j in $\tilde{S}$. We have $\tilde{S}_j \leq u_\ell + (\varepsilon u_q + 3\varepsilon^2 u_q) + 3(q-2)\varepsilon^2 u_q = u_\ell + 4\varepsilon u_q$, since, on the one hand, the total processing time of the tiny jobs assigned to u_ℓ on M_k in $\tilde{S}$ is at most $\varepsilon u_q + 3\varepsilon^2 u_q$, and, on the other hand, for each $\nu < \ell$ the total processing time of the tiny jobs assigned to u_ν and M_k in $\tilde{S}$ is greater by at most $3\varepsilon^2 u_q$ than the same amount in S^* (Observation 7) and the huge job assignment is the same in $\tilde{S}$ and S^*. Therefore $L'_j(\tilde{S}) = \tilde{C}_j - d_j + D \leq u_\ell + 5\varepsilon u_q - d_j + D \leq u_\ell + 5\varepsilon u_q - d_{j^*} + D \leq L'_{j^*}(S^*) + 5\varepsilon u_q \leq L'_{\max}(S^*) + 5\varepsilon u_q$ follows.

Now suppose that j is scheduled at step 5. We will show that there exists a tiny job j^* such that $S^*_{j^*} \geq \tilde{S}_j - 5\varepsilon u_q$ with $d_{j^*} \leq d_j$. From this the proposition follows, since $0 < p_j, p_{j^*} \leq \varepsilon^2 u_q$ by definition. Let $\tilde{A}(t)$ denote the set of tiny jobs j' that are scheduled at step 5 such that $\tilde{S}_{j'} \geq t$, and $\tilde{B}(t) := \mathcal{S}' \setminus \tilde{A}(t)$. Likewise, let $A^*(t)$ denote the set of tiny jobs j' with $S^*_{j'} \geq t$, and $B^*(t) := \mathcal{S}' \setminus A^*(t)$.

Claim 5. *If $t \geq u_q$, then $p(\tilde{A}(t + 5\varepsilon u_q)) \leq p(A^*(t))$.*

Proof. See Appendix A. □

From the claim we deduce $p(\tilde{B}(\tilde{S}_j)) \geq p(B^*(\tilde{S}_j - 5\varepsilon u_q))$. It follows that there exists $j^* \in \{j\} \cup \tilde{B}(\tilde{S}_j)$ such that $j^* \in A^*(\tilde{S}_j - 5\varepsilon u_q)$. Since the tiny jobs are scheduled in EDD order in $\tilde{S}$, we have $d_{j^*} \leq d_j$, and we are done. □

Proof of Theorem 4. If we put together the above results we get that Algorithm C constructs a feasible schedule in polynomial time and the (modified) lateness of this schedule is at most $L'_{\max}(S^{best}) \leq (1+\varepsilon)L'_{\max}(\tilde{S}) \leq (1+\varepsilon)(L'_{\max}(S^*) + 6\varepsilon u_q) \leq (1+8\varepsilon)L'_{\max}(S^*)$ by Propositions 8 and 9. □

8. Conclusions, open questions

We have shown a nearly full picture of the approximability of $P|rm|C_{\max}$, see Table 1. Two interesting questions are still open. Is there a PTAS for $P|rm = 1|C_{\max}$ or not? Is there an FPTAS for $1|rm = 1, q = const|C_{\max}$ for any constant greater than 2?

Conveying some of the ideas of this paper to solve scheduling problems with resource-consuming jobs in practice is subject to future work, which may require to study other objective functions as well.

Appendix A

Proof of Proposition 2. Let $\mathcal{J}^a(\hat{x})$ be the subset of jobs with $\hat{x}_{j\ell k} = 1$ for some $v_\ell > u_q$ and $k \in \mathcal{M}$. We define a new solution $\tilde{x}$ in which those jobs in $\mathcal{J}^a(\hat{x})$

are reassigned to new time points (but to the same machine) and show that $C_{\max}(\tilde{x}) \leq C_{\max}(\hat{x})$. Let $\tilde{x} \in \{0,1\}^{\mathcal{J}\times\mathcal{T}\times\mathcal{M}}$ 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 $\tilde{x}_{j\ell k} = 1$ for $v_\ell = \max\{u_q, r_j\}$ and for a k such that $\exists \ell' : \hat{x}_{j\ell' k} = 1$, 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 $k \in \mathcal{M}$ and $\ell = 1, \dots, \tau$,

$$v_\ell + \sum_{j\in\mathcal{J}} \sum_{\nu=\ell}^{\tau} p_j \tilde{x}_{j\nu k} \leq v_\ell + \sum_{j\in\mathcal{J}} \sum_{\nu=\ell}^{\tau} p_j \hat{x}_{j\nu k}, \tag{21}$$

from which the claim follows. If $v_\ell \leq u_q$, the left and the right-hand sides in (21) 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 (21) is verified again. □

Proof of Claim 3. Note that, each machine is working between u_q and $C_{\max} - \varepsilon^2 p_{\mathrm{sum}}$. Since $\bar{x}^{small}$ is an optimal solution of (7)-(12) and according to Proposition 3 $\hat{x}^{small}$ is a feasible solution, thus $p(\{j \in \mathcal{S} : S_j^* \leq u_q\}) \leq p(K) + p(U)$, where K is the set of small jobs scheduled at Step 5(ii)b of algorithm A, therefore $P_b^{(u_q)}(S_{\mathcal{S}}^*) \leq P_b^{(u_q+2|\mathcal{T}^b|\varepsilon^2 p_{\mathrm{sum}})}(S_{\mathcal{S}}) + p(U)$ (Proposition 5). $P_b^{(u_q)}(S_{\mathcal{B}}^*) \leq P_b^{(u_q+2|\mathcal{T}^b|\varepsilon^2 p_{\mathrm{sum}})}(S_{\mathcal{B}})$ follows also from Proposition 5, thus $P_b^{(u_q)}(S^*) \leq P_b^{(u_q+2|\mathcal{T}^b|\varepsilon^2 p_{\mathrm{sum}})}(S) + p(U)$, which implies $P_a^{(u_q)}(S^*) \geq P_a^{(u_q+2|\mathcal{T}^b|\varepsilon^2 p_{\mathrm{sum}})}(S) - p(U)$. Let $P_{S^*} := P_a^{(u_q)}(S^*)$ and $P_S := P_a^{(u_q+2|\mathcal{T}^b|\varepsilon^2 p_{\mathrm{sum}})}(S)$.

Note that $C_{\max} \leq u_q + 2|\mathcal{T}^b|\varepsilon^2 p_{\mathrm{sum}} + P_S/m + \varepsilon^2 p_{\mathrm{sum}}$ (Observation 3), $C^*_{\max} \geq u_q + P_{S^*}/m$ and $P_S \leq P_{S^*} + p(U)$. From these, $C_{\max} \leq C^*_{\max} + 2|\mathcal{T}^b|\varepsilon^2 p_{\mathrm{sum}} + p(U)/m + \varepsilon^2 p_{\mathrm{sum}}$ follows. Since $p(U) \leq (|\mathcal{R}| + 1) \cdot |\mathcal{T}^b|\varepsilon^2 p_{\mathrm{sum}}$ (Proposition 4), thus $C_{\max} \leq (1 + ((2|\mathcal{T}^b| + 1)m + (|\mathcal{R}| + 1) \cdot |\mathcal{T}^b|)\varepsilon^2)C^*_{\max}$, therefore we have proved Claim 3. □

Proof of Observation 7. The first part follows from $p(\mathcal{J}^*_{\ell,k}) + 3\varepsilon^2 u_q > (g^*_{k,\ell} - 2)(\varepsilon^2 u_q) + 3\varepsilon^2 u_q = (g^*_{k,\ell} + 1)(\varepsilon^2 u_q) > p(\tilde{\mathcal{J}}_{\ell,k})$ (the first inequality follows from the choice of g^*, while the second from the construction of $\tilde{S}$). For the second part, let $\ell' \leq \ell$ denote the last period where the algorithm had to proceed with the next period, because there was not enough resource to schedule the next tiny job, but $\mathcal{M}^*_{\ell'} \neq \emptyset$. The huge jobs that are assigned to a time period until $u_{\ell'}$ in $\tilde{S}$ are scheduled before $u_{\ell'+1}$ in S^*, thus, since $p_j = a_j$ and S^* is feasible, $p(\cup_{\nu \leq \ell'} \tilde{\mathcal{J}}_\nu) \geq p(\cup_{\nu \leq \ell'} \mathcal{J}^*_\nu) - \varepsilon^2 u_q$ follows, because otherwise there would be enough resource to assign at least one more tiny job to $u_{\ell'}$ in $\tilde{S}$. According to the definition of ℓ' and the rules of Algorithm C, we have $p(\tilde{\mathcal{J}}_\nu) \geq p(\mathcal{J}^*_\nu)$ for each $\nu = \ell' + 1, \ldots, \ell$, thus the observation follows. □

Proof of Claim 4. Assume for a contradiction that there exists an $\ell \leq q$ and $j_1 \in \tilde{\mathcal{J}}_\ell$ such that

$$d_{j_1} = \min\{d_{j'} : j' \in \tilde{\mathcal{J}}_\ell\} = \min\{d_{j'} : j' \in \cup_{\nu \geq \ell} \tilde{\mathcal{J}}_\nu\} < \min\{d_{j'} : j' \in \cup_{\nu \geq \ell} \mathcal{J}^*_\nu\}, \tag{22}$$

where the second equation follows from the EDD scheduling of tiny jobs in $\tilde{S}$. Let $H := \{j' \in \mathcal{S}' : d_{j'} \leq d_{j_1}\}$. Let $\ell' < \ell$ be the largest index such that $\mathcal{M}^*_{\ell'} \neq \emptyset$. If there is no such ℓ', then the claim follows, since we have $\bigcup_{\nu<\ell} \tilde{\mathcal{J}}_\nu = \bigcup_{\nu<\ell} \mathcal{J}^*_\nu = \emptyset$ from the definition of $\tilde{S}$. Otherwise, for each $\nu = \ell' + 1, \ldots, \ell - 1$, since $\mathcal{M}^*_\nu = \emptyset$, we have $\mathcal{J}^*_\nu = \tilde{\mathcal{J}}_\nu = \emptyset$. Furthermore, from (22), it follows that all the jobs in H start before $u_{\ell'+1}$ in S^* by our indirect assumption. Therefore,

$$p(\cup_{\nu \leq \ell'} \tilde{\mathcal{J}}_\nu) < p(H) \leq p(\cup_{\nu \leq \ell'} \mathcal{J}^*_\nu),$$

where the first inequality follows from the fact that H comprises all the tiny jobs assigned to any time period $u_\nu < u_\ell$ in $\tilde{S}$, and j_1 as well, which is assigned to u_ℓ by definition. Hence, case (i) of the Observation 8 must hold for ℓ'. Thus, there was not enough resource to schedule all the tiny jobs in H before $u_{\ell'+1}$ in $\tilde{S}$. On the other hand, all the jobs in H are scheduled before $u_{\ell'+1}$ in S^*, thus the resource consumption of the tiny jobs starting before $u_{\ell'+1}$ in S^* is not smaller than that in $\tilde{S}$. Moreover, the huge job assignment of the two schedules before u_q is the same. Since S^* is feasible, this is a contradiction. □

Proof of Claim 5. Note that, if $t \geq u_q$ then the total processing time of the huge jobs in $[\max\{C^{part}_{\max}(k), u_q\}, t]$ on any M_k in S^* is at least the total processing time of the huge jobs in $[\max\{C^{part}_{\max}(k), u_q\}, t + 5\varepsilon u_q]$ on M_k in $\tilde{S}$, because $\tilde{S}_{j'} \geq S^*_{j'} + 5\varepsilon u_q$ if j' is huge and $S^*_{j'} \geq u_q$. Since $p(\tilde{A}(u_q)) \leq p(A^*(u_q)) + \varepsilon^2 u_q$ (apply Observation 7 to $\ell = q - 1$), and there is no gap before any tiny job on any machine M_k in $\tilde{S}$ after $\max\{C^{part}_{\max}(k), u_q\}$, the claim follows, because there is more time to schedule tiny jobs until $t + 5\varepsilon u_q$ in $\tilde{S}$ on any machine for any $t \geq u_q$ than until t in S^*. □

Appendix B, PTAS for $P|preassign, r_j|L_{\max}$

In this section we sketch how to extend the PTAS of Hall & Shmoys (1989) for parallel machine scheduling with release dates, due-dates and the maximum lateness objective ($P|r_j|L_{\max}$) with pre-assigned works on the machines. The jobs scheduled on a machine must succeed any pre-assigned work.

Hall and Shmoys propose an $(1 + \varepsilon)$-optimal outline scheme in which job sizes, release dates, and due-dates are rounded such that the schedules can be labeled with concise outlines, and there is an algorithm which given any outline ω for an instance I of the scheduling problem, delivers a feasible solution to I of value at most $(1 + \varepsilon)$ times the value of any feasible solutions to I labeled with ω.

All we have to do to take pre-assigned work into account is that we extend the outline scheme of Hall and Shmoys with machine ready times, which are time points when the machines finish the pre-assigned work. Suppose the largest of these time points is $w_{\max}$. We divide $w_{\max}$ by $\varepsilon/2$ and round each of the pre-assigned work sizes of the machines down to the nearest multiple of $2w_{\max}/\varepsilon$. Thus the number of distinct pre-assigned work sizes is $\varepsilon/2$, a constant independent of the number of jobs and machines. Then, we amend the machine configurations (from which outlines are built) with the possible rounded pre-assigned work sizes. Finally, the algorithm which determines a feasible solution from an outline must be modified such that it disregards all the outlines in which any job is scheduled on a machine before the corresponding rounded pre-assigned work size in the outline, and if the rounded pre-assigned work sizes of the outline do not match the real pre-assigned works of the machines.

Acknowledgments

The authors are grateful to the referees for comments that helped to significantly improve the paper. This work has been supported by the OTKA grant K112881, and by the GINOP-2.3.2-15-2016-00002 grant of the Ministry of National Economy of Hungary. The research of Tamás Kis has been supported by the János Bolyai research grant BO/00412/12/3 of the Hungarian Academy of Sciences.

Artigues, C., Demassey, S., & Néron, E. (2013). *Resource-constrained project scheduling: models, algorithms, extensions and applications*. John Wiley & Sons.

Belkaid, F., Maliki, F., Boudahri, F., & Sari, Z. (2012). A branch and bound algorithm to minimize makespan on identical parallel machines with consumable resources. In *Advances in Mechanical and Electronic Engineering* (pp. 217–221). Springer. doi:10.1007/978-3-642-31507-7_36.

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. doi:10.1016/j.ejor.2010.05.036.

Briskorn, D., Jaehn, F., & Pesch, E. (2013). Exact algorithms for inventory constrained scheduling on a single machine. *Journal of Scheduling*, *16*, 105–115. doi:10.1007/s10951-011-0261-x.

Carlier, J. (1984). *Problèmes d'ordonnancements à contraintes de ressources: algorithmes et complexité. Thèse d'état*. Université Paris 6.

Carlier, J., & Rinnooy Kan, A. H. G. (1982). Scheduling subject to nonrenewable resource constraints. *Operational Research Letters*, *1*, 52–55. doi:10.1016/0167-6377(82)90045-1.

Carrera, S., Ramdane-Cherif, W., & Portmann, M.-C. (2010). Scheduling supply chain node with fixed component arrivals and two partially flexible deliveries. In *5th International Conference on Management and Control of Production and Logistics-MCPL 2010* (p. 6). IFAC Publisher. doi:10.3182/20100908-3-PT-3007.00030.

Gács, P., & Lovász, L. (1981). Khachiyan's algorithm for linear programming. *Mathematical Programming Studies*, *14*, 61–81.

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. doi:10.1016/j.mathsocsci.2011.04.004.

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

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. doi:10.1016/S0167-5060(08)70356-X.

Grigoriev, A., Holthuijsen, M., & van de Klundert, J. (2005). Basic scheduling problems with raw material constraints. *Naval Research of Logistics*, *52*, 527–553. doi:10.1002/nav.20095.

Györgyi, P., & Kis, T. (2014). Approximation schemes for single machine scheduling with non-renewable resource constraints. *Journal of Scheduling*, *17*, 135–144. doi:10.1007/s10951-013-0346-9.

Györgyi, P., & Kis, T. (2015a). Reductions between scheduling problems with non-renewable resources and knapsack problems. *Theoretical Computer Science*, *565*, 63–76. doi:10.1016/j.tcs.2014.11.007.

Györgyi, P., & Kis, T. (2015b). Approximability of scheduling problems with resource consuming jobs. *Annals of Operations Research*, *235*, 319–336. doi:10.1007/s10479-015-1993-3.

Hall, L. A., & Shmoys, D. B. (1989). Approximation schemes for constrained scheduling problems. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science* (pp. 134–139). IEEE. doi:10.1109/SFCS.1989.63468.

Herr, O., & Goel, A. (2016). Minimising total tardiness for a single machine scheduling problem with family setups and resource constraints. *European Journal of Operational Research*, *248*, 123–135. doi:10.1016/j.ejor.2015.07.001.

Janiak, A., Janiak, W., & Lichtenstein, M. (2007). Resource management in machine scheduling problems: a survey. *Decision Making in Manufacturing and Services*, *1*, 59–89. doi:10.7494/dmms.2007.1.2.59.

Kis, T. (2015). Approximability of total weighted completion time with resource consuming jobs. *Operations Research Letters*, *43*, 595–598. doi:10.1016/j.orl.2015.09.004.

Laborie, P. (2003). Algorithms for propagating resource constraints in ai planning and scheduling: Existing approaches and new results. *Artificial Intelligence*, *143*, 151–188. doi:10.1016/S0004-3702(02)00362-4.

Morsy, E., & Pesch, E. (2015). Approximation algorithms for inventory constrained scheduling on a single machine. *Journal of Scheduling*, *18*, 645–653. doi:10.1007/s10951-015-0433-1.

Neumann, K., & Schwindt, C. (2003). Project scheduling with inventory constraints. *Mathematical Methods of Operations Research*, *56*, 513–533. doi:10.1007/s001860200251.

Shabtay, D., & Kaspi, M. (2006). Parallel machine scheduling with a convex resource consumption function. *European Journal of Operational Research*, *173*, 92–107. doi:10.1016/j.ejor.2004.12.008.

Sirdey, R., Carlier, J., Kerivin, H., & Nace, D. (2007). On a resource-constrained scheduling problem with application to distributed systems reconfiguration. *European Journal of Operational Research*, *183*, 546–563. doi:10.1016/j.ejor.2006.10.011.

Slowinski, R. (1984). Preemptive scheduling of independent jobs on parallel machines subject to financial constraints. *European Journal of Operational Research*, *15*, 366–373. doi:10.1016/0377-2217(84)90105-X.

Stadtler, H., & Kilger, C. (2008). *Supply Chain Management and Advanced Planning. Concepts, Models, Software, and Case Studies*. (4th ed.). Springer.

Toker, A., Kondakci, S., & Erkip, N. (1991). Scheduling under a non-renewable resource constraint. *Journal of the Operational Research Society*, *42*, 811–814. doi:10.2307/2583664.

Williamson, D. P., & Shmoys, D. B. (2011). *The design of approximation algorithms*. Cambridge university press.

Xie, J. (1997). Polynomial algorithms for single machine scheduling problems with financial constraints. *Operations Research Letters*, *21*, 39–42. doi:10.1016/S0167-6377(97)00007-2.

Yeh, W.-C., Chuang, M.-C., & Lee, W.-C. (2015). Uniform parallel machine scheduling with resource consumption constraint. *Applied Mathematical Modelling*, *39*, 2131–2138. doi:10.1016/j.apm.2014.10.012.