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

Approximation schemes for single machine scheduling with non-renewable resource constraints

Péter Györgyi · Tamás Kis

Received: date / Accepted: date

Abstract In this paper we discuss exact and approximation algorithms for scheduling a single machine with additional non-renewable resource constraints. Given the initial stock levels of some non-renewable resources (e.g. raw materials, fuel, money), and time points along with replenishment quantities, a set of resource consuming jobs has to be scheduled on the machine such that there are enough resources for starting each job, and the makespan is minimized. We show that the problem admits a pseudo polynomial time algorithm when the number of replenishments is not part of the input, and also present an FPTAS when there is only a single resource, and it is replenished only once. We also describe a PTAS for the problem with a constant number of replenishments.

Keywords Single machine scheduling, non-renewable resources, approximation schemes

1 Introduction

Machine scheduling with additional non-renewable resource constraints is an exciting field with enormous practical importance. In this setting jobs have to be scheduled on the machine(s), while also respecting the availability of some non-renewable resources, which are consumed by the jobs, but replenished over time from external sources. When a job is started, it consumes those resources required to its execution in given quantities. This implies that not only the machine must be free when starting a job, but also the required resources must be available in sufficient quantities. Assuming that initially the non-renewable resources do not suffice to perform all the jobs, but the additional shipments, which occur at known moments in time, and in known quantities, together provide enough resources to complete all of them, it is a non-trivial task to find an ordering of the jobs such that the maximum job completion time (or other common performance measure) is minimized, while respecting the resource constraints. This model has been described by Carlier [5] (chapters VII and VIII), and by Carlier and Rinnooy Kan [4] (the models in that paper involve precedence constraints, and no machines) in the early 80's, and since then it has been studied by several others.

As an application, consider a workshop where each job requires a set of raw materials specified by its "bill-of-materials", and external suppliers ship various raw materials over time to a production line which processes the jobs. The jobs may share some of the materials, i.e., there is a "competition" between the jobs for the resources. The scheduler has to find an ordering of the jobs such that the idle time of the production line, due to waiting for material shipments, is minimized, or alternatively, when the jobs have due-dates, they complete on time as much as possible.

The $\alpha|\beta|\gamma$ notation of Graham et al. [11] has been extended with renewable resource constraints by Blazewicz et al. [1]. In addition, Grigoriev et al. [12] introduces the restrictions rm, and ddc, where $rm = m$ means that there are m raw materials (or non-renewable resources in general), and ddc indicates different dedicated raw materials (non-renewable resources) for each job. The initial availability or stock of the non-renewable resource(s) may be augmented by replenishments in distinct moments of time. When the number of replenishments is a fixed constant, then we add the restriction $q = const$ to the β field.

Scheduling with non-renewable resource constraints has been studied e.g. in [4], [5], [6], [9], [12], [13],[14], [15]. In particular, Carlier and Rinnooy Kan [4] study the problem with precedence constraints, but without machines, and derive polynomial time algorithms for various special cases. Carlier [5] establishes algorithmic and complexity results for several variants. Slowinski [14] consider a preemptive scheduling problem on parallel unrelated machines, and with some renewable resources, and one non-renewable resource (money) which becomes available in specified amounts at dif-

P. Györgyi
Department of Operations Research, Loránd Eötvös University, H1117 Budapest, Pázmány Péter sétány 1/C
T. Kis
Computer and Automation Research Institute, Hungarian Academy of Sciences, H1111 Budapest, Kende str. 13–17, Hungary
Tel.: +36 1 2796156; Fax: +36 1 4667503
E-mail: gyorgyipeter@gmail.com, tamas.kis@sztaki.mta.hu

ferent dates. Polynomial algorithms are developed for minimizing the schedule length, or total cost. Toker et al. [15] prove 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 at each time period. Grigoriev et al. [12] derive basic complexity results for several special cases of scheduling a single machine with non-renewable resource constraints, and propose some simple approximation algorithms for selected problems. Gafarov et al. [9] complement the findings of Grigoriev et al. by additional complexity results. Neumann and Schwindt [13] 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 [2], Briskorn et al. study the complexity of several variants, while Briskorn et al. [3] devise a branch-and-bound method for minimizing the weighted sum of job completion times on a single machine. However, none of these papers propose approximation schemes for NP-hard special cases of single machine scheduling subject to non-renewable resource constraints for the makespan objective.

Before summarizing our results, we recall the definition of a PTAS and an FPTAS, see e.g. the famous guide of Garey and Johnson [10], page 137. A *Polynomial Time Approximation Scheme* (PTAS) for an optimization problem is a family of algorithms $\{A_\varepsilon\}_{\varepsilon>0}$ such that A_ε has polynomial time complexity in the input length for each fixed $\varepsilon > 0$, and always delivers a solution which is $1+\varepsilon$ times the optimum value for a minimization problem, or at least $1-\varepsilon$ times the optimum for a maximization problem. A *Fully Polynomial Time Approximation Scheme* (FPTAS) is a family of algorithms $\{A_\varepsilon\}_{\varepsilon>0}$ with the same properties as a PTAS, plus each A_ε runs in polynomial time in $1/\varepsilon$ as well.

Results of this paper. We define the general problem $1|rm, ddc|C_{\max}$ formally in Section 2 and also discuss its computational complexity. Then we develop a pseudo polynomial time algorithm for solving the NP-hard $1|rm, ddc, q = const|C_{\max}$ special case to optimality in Section 3. For the still NP-hard special case $1|rm = 1, q = 2|C_{\max}$ we propose a fully polynomial time approximation scheme (Section 4). Moreover, for the problem $1|rm = 1, q = const|C_{\max}$ with a fixed number of replenishments we describe a polynomial time approximation scheme (Section 5). The PTAS can be extended to the $1|rm, ddc - agr, q = const|C_{\max}$ problem in which there are different resources dedicated to each job, and the resource requirements of the jobs are agreeable, i.e., there is a total order of the jobs which agrees with a total ordering of the resource requirements of the jobs for each resource (Section 6).

This paper is a follow up to Drótos and Kis [8] in which the scheduling of inventory releasing jobs has been studied.

2 Problem $1|rm, ddc|C_{\max}$

Given a set of jobs $\mathcal{J}$, a set of non-renewable resources $\mathcal{R}$, and a single machine. Each job J_j has a processing time $p_j \in \mathbb{Z}_+$, and resource requirements specified by a non-negative vector $a_j \in \mathbb{Z}_0^{\mathcal{R}}$. The resources are supplied in q distinct moments in time, $\tau_1, \ldots, \tau_q \in \mathbb{Z}_0$, where $\tau_1 = 0$, and $\tau_\ell < \tau_{\ell+1}$ for $1 \le \ell \le q-1$. The quantity of the resources supplied at τ_ℓ is specified by an $|\mathcal{R}|$-dimensional vector $\tilde{b}_\ell \in \mathbb{Z}_0^{\mathcal{R}}$. All problem data is integral. When a job is started, it immediately decreases the inventory of the resources by its requirements. A schedule Sch specifies the starting time s_j of each job J_j. A schedule is *feasible* if and only if

1. the processing of jobs do not overlap in time, and
2. for any time point t, the total resource requirements of those jobs started up to time t do not exceed the total supply up to time t.

The objective is to minimize the completion time of the job finished last.

For the sake of simpler presentation, we define one more moment in time, $\tau_{q+1} := \tau_q + \sum_{j\in\mathcal{J}} p_j$, which marks the end of the scheduling time horizon (there is no material supply in τ_{q+1}). The intervals $[\tau_\ell, \tau_{\ell+1})$ are called *supply periods*, $\ell = 1, \ldots, q$. The total resource supply over the first ℓ supply events is $b_\ell := \sum_{u=1}^{\ell} \tilde{b}_u$. Throughout the paper we assume that the total resource supply (over the q supply periods) is sufficient to process all the jobs, but it is insufficient over the first $q-1$ supply periods, i.e., $b_q \ge \sum_{j\in\mathcal{J}} a_j$ (componentwise), and there exists a resource $i \in \mathcal{R}$ such that $b_{q-1}(i) < \sum_{j\in\mathcal{J}} a_j(i)$. This implies that scheduling all the jobs in supply period q yields a feasible schedule, and in any feasible schedule at least one job is assigned to the last supply period, whence the optimal makespan $C^*_{\max}$ is greater than τ_q.

We can model the above scheduling problem by a mathematical program. There are $q|\mathcal{J}|$ decision variables $x_{\ell j}$ representing the assignment of jobs to supply periods, i.e., $x_{\ell j} = 1$ if and only if the resource requirements of job J_j must be satisfied from the total resource supply over the first ℓ supply periods, and it does not

start before τ_ℓ:

$$C^*_{\max} = \min \max_{\ell=1,\dots,q} \left(\tau_\ell + \sum_{u=\ell}^{q} \sum_{j \in J} p_j x_{uj} \right) \quad (1)$$

s.t.

$$\sum_{j \in \mathcal{J}} a_j \left(\sum_{u=1}^{\ell} x_{uj} \right) \le b_\ell, \quad \ell = 1, \dots, q-1 \quad (2)$$

$$\sum_{\ell=1}^{q} x_{\ell j} = 1, \quad j \in \mathcal{J} \quad (3)$$

$$x_{\ell j} \in \{0,1\}, \quad \ell = 1, \dots, q, \ j \in \mathcal{J} \quad (4)$$

The objective function (1) expresses that we want to minimize the maximum job completion time. Constraints (2) express that the jobs assigned to the first ℓ supply periods cannot consume more resource(s) than the total supply over the first ℓ supply periods. Constraints (3) ensure that each job has to be assigned to exactly one supply period.

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 supply period. That is, sequence the jobs assigned to supply period ℓ in some order. Let these sequences be σ_ℓ, $\ell = 1, \dots, q$, and join the pieces in increasing order of the supply periods, i.e., $(\sigma_1, \sigma_2, \dots, \sigma_q)$. Let S_ℓ and C_ℓ denote the starting time and completion time of the piece σ_ℓ, respectively. Then we have $C_\ell = S_\ell + \sum_{j \in \mathcal{J}} p_j \bar{x}_{\ell j}$, $S_1 = \tau_1 = 0$, and $S_\ell = \max\{\tau_\ell, C_{\ell-1}\}$ for $\ell = 2, \dots, q$. Notice that the S_ℓ and C_ℓ are uniquely determined by $\bar{x}$.

The problem with one non-renewable resource and a single machine has been introduced by Carlier [5], and he has also shown that it is NP-hard in the ordinary sense for $q = 2$ supply periods by a reduction from the PARTITION problem ([5], Proposition 5 of Section 4.2.1).

Lemma 1 *[5] Problem* $1|rm = 1, q = 2|C_{\max}$ *is NP-hard in the ordinary sense.*

Carlier has also proved that for general q, the problem is NP-had in the strong sense ([5], Proposition 6 of Section 4.2.1). This has been also observed by Grigoriev et al. [12].

We close this section by summarizing the notation used throughout the paper (Table 1) and with additional terminology. The resource requirements of the jobs are *agreeable*, denoted by $ddc - agr$ in the β field, if there is a sequence of jobs ω such that for $j < k$, $a_{\omega(j)}(i) \le a_{\omega(k)}(i)$ for each $i \in \mathcal{R}$, and $\omega(j)$ and $\omega(k)$ are the jobs in positions j and k, respectively, of sequence ω.

Table 1 Notation

$n^{\mathcal{J}}$	number of jobs
$\mathcal{J}$	set of jobs $\{J_1, \dots, J_{n^{\mathcal{J}}}\}$
p_j	processing time of job J_j
$p_{\max}$	maximum processing time of the jobs, i.e., $p_{\max} = \max p_j$
p_{sum}	total processing time of the jobs, i.e., $p_{\text{sum}} = \sum p_j$
$\mathcal{R}$	set of resources
a_j	resource requirements of job J_j
a_{sum}	total resource requirements of the jobs
q	number of moments in time when some resource is supplied
τ_ℓ	the ℓ^{th} time moment when some resource is supplied, $0 = \tau_1 < \tau_2 < \dots < \tau_q$
$\tilde{b}_\ell$	vector of resource supplies at time moment τ_ℓ
b_ℓ	total resource supply up to time moment τ_ℓ, i.e., $b_\ell = \sum_{u=1}^{\ell} \tilde{b}_u$

3 A pseudo polynomial time algorithm for $1|rm = const, ddc, q = const|C_{\max}$

We reduce the problem $1|rm = const, ddc, q = const|C_{\max}$ to finding a path in an acyclic digraph from the source node to a terminal node representing a solution with smallest objective function value. The nodes of the graph represent the total time, and resource consumption, respectively, of the jobs (already) scheduled in each of the q supply periods, while the arcs indicate the assignment of jobs to supply periods. The directed paths in the graph correspond to schedules. Although there are exponentially many directed paths (or schedules), but the number of nodes (and arcs) remains pseudo polynomial, since the total time or resource consumption can be bounded by adding up the respective problem data. The graph consists of $n^{\mathcal{J}} + 1$ layers: the 0. layer contains the unique source node, and we define the nodes on the other layers iteratively, along with the arcs. A node at layer i represents an assignment of the first j jobs to the supply periods, i.e., node $N_j(P_1, \dots, P_q, \Delta^1, \dots, \Delta^q)$ represents an assignment of the first j jobs in which the total processing time of those jobs assigned to supply period ℓ is P_ℓ, and the total demand from the $|\mathcal{R}|$ resources in supply period ℓ is $\Delta^\ell \in \mathbb{Z}_0^{\mathcal{R}}$, for $1 \le \ell \le q$. Notice that the source node is $N_0(0, \dots, 0)$. Now from any node $N_j(P_1, \dots, P_q, \Delta^1, \dots, \Delta^q)$ of layer $j < n^{\mathcal{J}}$, we direct arcs to q nodes of layer $j+1$, i.e., we assign job J_{j+1} to each supply period in turn. For supply period ℓ, we direct an arc to node $N_{j+1}(P_1, \dots, P_\ell + p_{j+1}, \dots, P_q, \Delta^1, \dots, \Delta^\ell + a_{j+1}, \dots, \Delta^q)$, and label the arc with job J_{j+1}. Those nodes at layer $n^{\mathcal{J}}$ reachable from the source node on a directed path are called *terminal nodes*. We evaluate every terminal node to find the one representing a solution of minimum objective

function value. Namely, we check whether $\sum_{u=1}^{\ell} \Delta^u \leq b_u$ for $\ell = 1, \ldots, q-1$ (the inequality holds for $\ell = q$ by definition). If a terminal node satisfies the conditions, then any directed path from the source to the terminal node represents a feasible solution of (1)-(4).

The *assignment* corresponding to a directed path π from the source node to a terminal node is defined as $x^{\pi}_{\ell j} = 1$ if and only if job j is assigned to supply period ℓ on π.

The objective function value of a terminal node is $\max_{\ell} \tau_\ell + \sum_{\kappa=\ell}^{q} P_\kappa$, which is the makespan of the corresponding schedule. Now observe that for any node $N_j(P_1, \ldots, P_q, \Delta^1, \ldots, \Delta^q)$ of the graph constructed in the course of the algorithm, $0 \leq P_u \leq \sum_{k=1}^{j} p_k \leq p_{\text{sum}}$, and $0 \leq \Delta^u(i) \leq \sum_{k=1}^{j} a_k(i) \leq a_{\text{sum}}(i)$, $i \in \mathcal{R}$, hold for each $u = 1, \ldots, q$. Since both q and the number resources are constant, the total number of nodes is pseudo polynomial in the input. Since each node has a maximum degree of q, the same holds for the number of arcs. Therefore, we have shown the following:

Theorem 1 *The problem* $1|rm = const, ddc, q = const|C_{\max}$ *can be solved in pseudo polynomial time.*

We have implemented the pseudo polynomial time algorithm in C++ programming language to assess its practical complexity. We have generated test instances by varying the number of jobs, the number of supply periods, the ratio of the material supply in the supply periods, and the maximum processing time and resource requirement. We have generated data with one resource, and $n \in \{25, 50\}$ jobs, respectively. The processing times and resource requests of the jobs were chosen uniformly at random between 1 and $p_{\max}$, and 1 and $a_{\max}$, respectively, with $p_{\max} = a_{\max} \in \{5, 10\}$. The resource supplies $\tilde{b}_\ell$, $\ell = 1, \ldots, q$, were determined by tuples $(x_1, \ldots, x_q)$, such that $\sum_{u=1}^{q} x_u = 1$, and $\tilde{b}_\ell = x_\ell \sum_{u=1}^{q} a_u$, for $\ell \in \{1, \ldots, q\}$. For 25 jobs, we generated instances with $q = 2$ supply periods, and three tuples $\{(0.5, 0.5), (0.25, 0.75), (0.2, 0.8)\}$, and also instances with $q = 3$ supply periods, and two tuples $\{(1/3, 1/3, 1/3), (0.2, 0.2, 0.6)\}$. For 50 jobs, we considered only $q = 2$ supply periods, and three tuples $\{(0.5, 0.5), (0.25, 0.75), (0.2, 0.8)\}$. For each combination of parameter settings, we generated 5 random instances. The results are summarized in Table 2. Each cell of the table indicates the average number of graph nodes generated when solving the instances with the corresponding parameter settings. The computation times on instances with 2 supply periods were in the order of seconds, whereas on instances with 3 supply periods, and $a_{\max} = p_{\max} = 5$ in the order of minutes. We have results neither with 50 jobs and 3 supply periods, nor with 25 jobs, 3 supply periods when $p_{\max} = a_{\max} = 10$, because the computational time on such instances was too high (more than 60 minutes). Observe that the more supply is left to the last supply period, the less nodes the graphs have on average, which is plausible as fewer jobs can be assigned to the first period. All in all, the algorithm is not sophisticated enough for solving practical problems, but it suffices for building a fully polynomial time approximation scheme on it in the next section.

4 An FPTAS for $1|rm = 1, q = 2|C_{\max}$

In this section we describe an FPTAS for the special case with one resource and two supply periods.

To this end, we will round both the processing times and the resource requirements of the jobs. Namely, let $K = \varepsilon p_{\max}/n^{\mathcal{J}}$, and $L = \varepsilon a_{\max}$. Then we define

$$p_j^{\#} = K\lfloor p_j/K \rfloor, \quad a_j^{\#} = L\lfloor a_j/L \rfloor, \qquad \forall\, j. \tag{5}$$

We build a directed graph with $n^{\mathcal{J}} + 1$ layers similarly as in Section 3 using the rounded processing times and resource requirements, and we label the arcs with the jobs, and also with weights as follows: if the arc assigns some job J_j to the first period, then the weight is $a_j - a_j^{\#}$, otherwise it is 0. The nodes of this graph are denoted by $N_i(P_1^{\#}, P_2^{\#}, \Delta_1^{\#}, \Delta_2^{\#})$, where i identifies the layer, and $P_\ell^{\#}$, and $\Delta_\ell^{\#}$ represent the total rounded processing time and total rounded resource requirement of those jobs assigned to supply period $\ell = 1, 2$, respectively. At layer 0 there is a unique source node $N_0(0, 0, 0, 0)$. From each node from layer $0 \leq i < n^{\mathcal{J}}$ there are two arcs directed to two nodes at layer $i + 1$, one arc assigns job J_{i+1} to the first supply period, and this arc has a weight of $a_j - a_j^{\#}$, and the other arc assigns the job to the second supply period, and has a weight of 0. Those nodes at layer $n^{\mathcal{J}}$ reachable from the source node on a directed path are called *terminal nodes*.

We say that a terminal node *represents a feasible solution*, if there is a directed path from the source node leading to the terminal node such that the path represents a feasible assignment $\bar{x}$ of jobs to supply periods, i.e., $\bar{x}$ satisfies (2).

Lemma 2 *A terminal node* $N_{n^{\mathcal{J}}}(P_1^{\#}, P_2^{\#}, \Delta_1^{\#}, \Delta_2^{\#})$ *represents a feasible solution if and only if the minimum weight* w^* *of those paths from the source node leading to this node satisfies the condition* $\Delta_1^{\#} + w^* \leq b_1$.

Proof First suppose there is a directed path from the source node to node $N_{n^{\mathcal{J}}}(P_1^{\#}, P_2^{\#}, \Delta_1^{\#}, \Delta_2^{\#})$ which represents an assignment $\bar{x}$ of jobs to the supply periods with $\sum_j a_j \bar{x}_{1,j} \leq b_1$. Let w denote the weight of this

	$p_{\max} = a_{\max} = 5$				$p_{\max} = a_{\max} = 10$			
	$q = 2$		$q = 3$		$q = 2$		$q = 3$	
$n = 25$	$(0.5, 0.5)$	12100	$(1/3, 1/3, 1/3)$	3426513	$(0.5, 0.5)$	39335	$(1/3, 1/3, 1/3)$	n.a.
	$(0.25, 0.75)$	5283	$(0.2, 0.2, 0.6)$	918463	$(0.25, 0.75)$	15163	$(0.2, 0.2, 0.6)$	n.a.
	$(0.2, 0.8)$	3744			$(0.2,\ 0.8)$	10486		
$n = 50$	$(0.5, 0.5)$	102000	$(1/3, 1/3, 1/3)$	n.a.	$(0.5, 0.5)$	341000	$(1/3, 1/3, 1/3)$	n.a.
	$(0.25, 0.75)$	45001	$(0.2, 0.2, 0.6)$	n.a.	$(0.25, 0.75)$	153172	$(0.2, 0.2, 0.6)$	n.a.
	$(0.2, 0.8)$	30945			$(0.2, 0.8)$	113116		

Table 2 Computational results with the pseudo polynomial algorithm.

path. Then we have $\sum_j a_j \bar{x}_{1,j} = w + \Delta_1^\#$. Since w^* represents the minimum weight of a directed path from the source node to $N_{n^\mathcal{J}}(P_1^\#, P_2^\#, \Delta_1^\#, \Delta_2^\#)$, we have $w^* \leq w$. Therefore, $\Delta_1^\# + w^* \leq b_1$.

Conversely, suppose $\Delta_1^\# + w^* \leq b_1$ for the minimum weight w^* of a directed path from the source node to $N_{n^\mathcal{J}}(P_1^\#, P_2^\#, \Delta_1^\#, \Delta_2^\#)$. Let $\bar{x}$ be the assignment of jobs to supply periods represented by this shortest path. Then we have $\Delta_1^\# = \sum_j a_j^\# \bar{x}_{1,j}$. Consequently, $b_1 \geq \Delta_1^\# + w^* = \sum_j a_j^\# \bar{x}_{1,j} + w^* = \sum_j a_j \bar{x}_{1,j}$, and the claim follows. □

The *value* of a terminal node is $\max_{\ell=1,2} \tau_\ell + \sum_{\kappa=\ell}^{2} \sum_{j \in \mathcal{J}} p_j^\# \bar{x}_{\kappa j}$, where $\bar{x}$ is the assignment corresponding to the smallest weight path from the source node to the terminal node.

Lemma 3 *A feasible terminal node with smallest value represents a solution $\bar{x}$ of the scheduling problem of makespan at most $C^*_{\max}(1 + \varepsilon)$.*

Proof We pick a terminal node $N_{n^\mathcal{J}}(P_1^\#, P_2^\#, \Delta_1^\#, \Delta_2^\#)$ as given in the statement of the lemma. Then we clearly have $\max\{P_1^\# + P_2^\#, \tau_2 + P_2^\#\} \leq C^*_{\max}$, since $p_j^\# \leq p_j$ for all $j \in \mathcal{J}$. Since $p_j \leq p_j^\# + \varepsilon p_{\max}/n^\mathcal{J}$, we have

$$\max_{\ell=1,2} \tau_\ell + \sum_{u=\ell}^{2} \sum_{j \in \mathcal{J}} p_j \bar{x}_{uj} \leq$$

$$\max_{\ell=1,2} \tau_\ell + \sum_{u=\ell}^{2} \sum_{j \in \mathcal{J}} p_j^\# \bar{x}_{uj} + \varepsilon p_{\max} \leq C^*_{\max}(1 + \varepsilon).$$

□

Algorithm A:

1. Construct the layered directed graph.
2. Find a terminal node which represents a feasible solution and has the smallest value.
3. Output the best feasible solution found in the second step.

Theorem 2 *Algorithm A is indeed an FPTAS for $1|rm = 1, q = 2|C_{\max}$.*

Proof Since in step 2 all the feasible terminal nodes are evaluated and the one with smallest value is selected, Lemma 3 implies that the output of the algorithm is at most $(1 + \varepsilon)$ times the optimum.

The running time of the algorithm is dominated by the construction of the directed graph. Since the rounded job processing times are of the form $K \cdot k$ for some $0 \leq k \leq \lfloor n^\mathcal{J}/\varepsilon \rfloor$, and the rounded resource requirements are of the form $L \cdot \ell$ for some $0 \leq \ell \leq \lfloor 1/\varepsilon \rfloor$, the number of nodes is $O(n^\mathcal{J}(n^\mathcal{J}\lceil n^\mathcal{J}/\varepsilon \rceil)^2(n^J\lceil 1/\varepsilon \rceil)^2) = O((n^\mathcal{J})^7\varepsilon^{-4})$. Since there are two arcs emanating from each node, the number of arcs is of the same order. Hence, the size of the graph is polynomial in $n^\mathcal{J}$ and $1/\varepsilon$. Since the construction of the graph is polynomial in its size, and finding the best terminal node is also polynomial in the number of terminal nodes and in $n^\mathcal{J}$, the entire procedure is polynomial in $n^\mathcal{J}$ and $1/\varepsilon$. Hence, the algorithm is indeed an FPTAS. □

5 A PTAS for $1|rm = 1, q = const|C_{\max}$

In this section we describe a PTAS for the problem $1|rm = 1, q = const|C_{\max}$. Suppose we want to achieve an error ratio $1 + \rho$, where $\rho > 0$ is fixed. We will describe a procedure which for any fixed parameter $\varepsilon > 0$ always delivers a solution of value at most $(1 + c \cdot \varepsilon)$ times the optimum, where the constant $c > 0$ does not depend on the input, or on ε. Therefore, to obtain an algorithm with an error ratio of $1 + \rho$, we choose ε such that $0 < \varepsilon \leq \rho/c$ holds. To simplify the presentation, we also assume that $1/\varepsilon$ is integral, so we may let $\varepsilon = 1/\lceil c/\rho \rceil$.

A job is *big* if $p_j \geq \varepsilon p_{\text{sum}}$, otherwise it is small. Let $\mathcal{B}$ be the set of big jobs, and $\mathcal{S}$ the set of small jobs. The main idea is that we first assign the big jobs to supply periods in all possible ways, and then we complete each assignment by inserting the small jobs into the schedule in a suboptimal way using an approximation algorithm for a special knapsack problem. Finally, we choose the best schedule obtained. An *assignment* of jobs to supply periods is a binary vector $\bar{x} \in \{0, 1\}^{n \times q}$, where $\bar{x}_{\ell j} = 1$ if and only if job J_j is assigned to supply period ℓ. An assignment $\bar{x}$ can be separated into an assignment $\bar{x}^B$

of big jobs to supply periods, and an assignment $\bar{x}^S$ of the small jobs to supply periods, i.e., $\bar{x} = (\bar{x}^B, \bar{x}^S)$. An assignment $\bar{x}^B$ of big jobs to supply periods is *eligible* if and only if the following condition is satisfied: for each ℓ in $1, \ldots, q-1$: $\sum_{u=1}^{\ell} \sum_{j \in \mathcal{B}} a_j \bar{x}^B_{uj} \leq b_\ell$. Clearly, eligibility means that the resource constraints are not violated by the assignment $\bar{x}^B$ of big jobs to supply periods.

Recall the mathematical programming formulation (1)-(4). In the following we will frequently use the restricted version of this mathematical program when the assignment of big jobs is fixed, i.e., x^B is set to some eligible assignment $\bar{x}^B$ of big jobs to supply periods. Let $IP(\bar{x}^B)$ denote the resulting mathematical program, and $\mathrm{OPT}(\bar{x}^B)$ its optimum value. For the fixed $\bar{x}^B$, define a schedule of big jobs as follows: $\bar{C}^B_\ell = \max\{\bar{C}^B_{\ell-1}, \tau_\ell\} + \sum_{j \in \mathcal{B}} p_j \bar{x}^B_{\ell j}$ for $\ell = 1, \ldots, q$, where $\bar{C}^B_0 = 0$. The PTAS presented in this section relies on the following structural property of $IP(\bar{x}^B)$.

Lemma 4 *The mathematical program $IP(\bar{x}^B)$ admits an optimal solution $\bar{x}^S$ such that*

(i) if $\bar{C}^B_\ell \geq \tau_{\ell+1}$ then no small job is assigned to supply period ℓ,

(ii) the total processing time of those small jobs assigned to supply period ℓ is at most $\tau_{\ell+1} + \varepsilon p_{\mathrm{sum}} - \bar{C}^B_\ell$.

Proof Let $\bar{x}^S$ be an optimal solution of $IP(\bar{x}^B)$. Choose any schedule corresponding to $(\bar{x}^B, \bar{x}^S)$ in which, without loss of generality, for each supply period ℓ, the big jobs assigned to ℓ precede the small ones assigned to the same supply period. We may even assume that the small jobs assigned to a supply period are in shortest processing time order, which ensures that the last small job assigned to a supply period starts at the earliest possible time. Now if the last small job assigned to a supply period ℓ actually starts at $\tau_{\ell+1}$ or later in the schedule, then we reassign it to the next supply period, and reinsert it into the schedule. Namely, let ℓ be the first supply period such that there is a small job $j \in \mathcal{S}$ with $\bar{x}^S_{\ell,j} = 1$, but j does not start before $\tau_{\ell+1}$ in the schedule. Then the machine is not idle in the entire supply period ℓ. We reassign j to supply period $\ell + 1$, and reinsert it into the schedule of jobs, if any, already assigned to supply period $\ell + 1$. Clearly, this update of the schedule does not increase the makespan. We repeat the reinsertion and reassignment of small jobs until no small job assigned to some supply period ℓ, but starting not before $\tau_{\ell+1}$ exists. Since the big jobs assigned to supply period ℓ do not finish before $\bar{C}^B_\ell$, condition (i) is satisfied by the updated $\bar{x}^S$.

Finally, since the length of a small job is at most $\varepsilon p_{\mathrm{sum}}$, and the big jobs assigned to supply period ℓ do not finish before $\bar{C}^B_\ell$, all the small jobs assigned to supply period ℓ finish by $\tau_{\ell+1} + \varepsilon p_{\mathrm{sum}}$, which implies (ii). □

The *value* $v(\bar{x})$ of an assignment $\bar{x}$ of jobs to supply periods is defined as the objective function value (1) for $x = \bar{x}$, provided $\bar{x}$ satisfies (2)-(4), and $v(\bar{x}) = +\infty$ otherwise.

Algorithm B:

1. Assign the big jobs in all possible ways to the q supply periods. For each assignment $\bar{x}^B$ perform the steps 2-3 as follows:
2. If $\bar{x}^B$ is not eligible, drop this assignment, and consider the next assignment of big jobs.
3. Determine a schedule of big jobs, i.e., let $\bar{C}^B_0 = 0$, and $\bar{C}^B_\ell = \max\{\bar{C}^B_{\ell-1}, \tau_\ell\} + \sum_{j \in \mathcal{B}} p_j \bar{x}^B_{\ell j}$ for $\ell = 1, \ldots, q$. Let $\bar{b}_\ell = b_\ell - \sum_{j \in \mathcal{B}} a_j \left(\sum_{u=1}^{\ell} \bar{x}^B_{uj} \right)$. Since $\bar{x}^B$ is eligible, $\bar{b}_\ell \geq 0$ for all $\ell = 1, \ldots, q$. Define the following mathematical program:

$$\mathrm{OPT}^S_{\bar{x}^B} := \max \sum_{\ell=1}^{q-1} \sum_{j \in \mathcal{S}} p_j x_{\ell j} \tag{6}$$

s.t.

$$\sum_{j \in \mathcal{S}} a_j \left(\sum_{u=1}^{\ell} x_{uj} \right) \leq \bar{b}_\ell, \quad \ell = 1, \ldots, q-1 \tag{7}$$

$$\sum_{j \in \mathcal{S}} p_j x_{\ell j} \leq \max\{0, \tau_{\ell+1} - \bar{C}^B_\ell\} + \varepsilon p_{\mathrm{sum}}, \quad \ell = 1, \ldots, q-1 \tag{8}$$

$$\sum_{u=1}^{q-1} x_{uj} \leq 1, \quad j \in \mathcal{J} \tag{9}$$

$$x_{uj} \in \{0, 1\}, \ j \in \mathcal{J} \tag{10}$$

Let $p^S_{\mathrm{sum}} = \sum_{j \in \mathcal{S}} p_j$. Find an ε-approximate solution $\hat{x}^S$ of this program such that $\sum_{u=1}^{q-1} \sum_{j \in \mathcal{S}} p_j \hat{x}_{uj} \geq (1 - O(\varepsilon))\mathrm{OPT}^S_{\bar{x}^B}$, which may even violate the constraints (8) by an amount of $\varepsilon p^S_{\mathrm{sum}}$ in total. Compute $v((\bar{x}^B, \hat{x}^S))$.
4. Output the best solution obtained.

Firstly, we prove that for any eligible assignment $\bar{x}^B$ of the big jobs to supply periods, if $\hat{x}^S$ is an ε-approximate solution, then the value of the assignment $\hat{x} = (\bar{x}^B, \hat{x}^S)$ is a good approximation of the optimal solution of (1)-(4) with x^B fixed to $\bar{x}^B$.

Lemma 5 *Let $\hat{x}^S$ be an ε-approximate solution of (6)-(10), which may violate the constraints (8) by an*

amount of $\varepsilon p^S_{\rm sum}$ *in total. Then* $v(\hat{x})$*, the value of the assignment* $\hat{x} = (\bar{x}^B, \hat{x}^S)$*, is at most* $(1+O(\varepsilon))\mathrm{OPT}(\bar{x}^B)$*.*

Proof Let $\tilde{x}^S$ be an optimal solution of $IP(\bar{x}^B)$, which, without loss of generality, satisfies the conditions of Lemma 4. Hence, $\tilde{x}^S$ is a feasible solution of (6)-(10). Therefore, $\sum_{j\in\mathcal{S}} p_j \left(\sum_{\ell=1}^{q-1} \tilde{x}^S_{\ell j}\right) \leq \mathrm{OPT}^S_{\bar{x}^B}$ (the optimum value of (6)-(10)). Therefore, in order to approximate $\mathrm{OPT}(\bar{x}^B)$, we need a solution which assigns small jobs to supply periods $1,\ldots,q-1$ of total processing time close to $\mathrm{OPT}^S_{\bar{x}^B}$.

Now let $\hat{x}^S$ be an ε-approximate solution of (6)-(10) which may violate (8) by an amount of $\varepsilon p^S_{\rm sum}$ in total. Since $\sum_{j\in\mathcal{S}} p_j \left(\sum_{\ell=1}^{q-1} \hat{x}_{\ell j}\right) \geq (1-O(\varepsilon))\mathrm{OPT}^S_{\bar{x}^B}$, and the constraints (8) may be violated by at most $\varepsilon p^S_{\rm sum}$ in total, the value of the assignment $(\bar{x}^B, \hat{x}^S)$ is by at most $(q-1)\varepsilon p_{\rm sum} + \varepsilon p^S_{\rm sum} + O(\varepsilon)\mathrm{OPT}^S_{\bar{x}^B}$ more than $\mathrm{OPT}(\bar{x}^B)$. Observe that $p^S_{\rm sum} \leq p_{\rm sum}$ and both $p_{\rm sum}$ and $\mathrm{OPT}^S_{\bar{x}^B}$ are lower bounds on $\mathrm{OPT}(\bar{x}^B)$. Hence,

$$v((\bar{x}^B,\hat{x}^S)) \leq \mathrm{OPT}(\bar{x}^B) + \varepsilon p^S_{\rm sum} + (q-1)\varepsilon p_{\rm sum} + O(\varepsilon)\mathrm{OPT}^S_{\bar{x}^B} \leq (1+O(\varepsilon))\mathrm{OPT}(\bar{x}^B).$$

□

Now we turn to finding an ε-approximate solution of (6)-(10). Our approach builds on the ideas of Chekuri and Khanna [7] who devised a PTAS for solving the Multiple Knapsack Problem (MKP). The MKP problem is as follows: Given a set $\mathcal{B}$ of m bins, and a set $\mathcal{S}$ of n items. Each bin $i \in \mathcal{B}$ has a capacity of $c(i)$, and each item $j \in \mathcal{S}$ has a size $e(j)$, and a profit $p(j)$. Find a subset $U \subseteq \mathcal{S}$ of items of maximum profit such that the items in U can be packed into the m bins. The PTAS of Chekuri and Khanna is divided into a guessing stage and a packing stage. In the guessing stage the optimum value is "guessed" along with a set of items which can be packed into to bins, whereas in the packing stage a subset of items is chosen and a feasible packing is sought while losing only an $O(\varepsilon)$ fraction of the optimum profit, where $\varepsilon > 0$ is a parameter.

The problem (6)-(10) has some similarities to MKP: we have to select a subset of small jobs of maximum total processing time (profit). The bins are the first $q-1$ supply periods with capacities $\max\{0, \tau_{i+1} - \bar{C}^B_i\} + \varepsilon p_{\rm sum}$. By Lemma 5, these capacity constraints may be violated by $\varepsilon p^S_{\rm sum}$ in total to obtain a solution which has a value of at most $(1+c\cdot\varepsilon)$ times the optimum, where c is a constant independent of ε and the input. Moreover, we have additional size parameters, the a_j values, and capacity constraints (7) of the bins, which cannot be violated. Notice that the additional capacity constraints are nested, which can be exploited when packing the items. Firstly, we will guess the true optimum value of (6)-(10), where guessing means that we define a set of possible values such that one of them is close enough to the true optimum, and whose number is polynomial in the length of the input. Then we will round the job processing times and partition the set of small jobs according to the rounded job processing times. We will also guess the total processing time of those small jobs assigned to each supply period from each subset of the partitioning. Finally, we will assign the jobs to the supply periods in increasing a_j order. We will show that all the guessing steps can be done in polynomial time in n, and that we get an ε-approximate solution in the end.

For a fixed $\bar{x}^B$, let $\mathcal{S}(\bar{x}^B)$ be the set of those small jobs which may be assigned to a supply period $\ell \leq q-1$, i.e., $a_j \leq \bar{b}_\ell$. Clearly, all the small jobs in $\mathcal{S} \setminus \mathcal{S}(\bar{x}^B)$ can only be assigned to supply period q in any feasible schedule. Let $n = |\mathcal{S}(\bar{x}^B)|$ denote the number of small jobs in $\mathcal{S}(\bar{x}^B)$, and $p^{S(\bar{x}^B)}_{\max} = \max_{j\in\mathcal{S}(\bar{x}^B)} p_j$. In addition, $\varepsilon > 0$ is a parameter determining the error of the algorithm, and to simplify notation, we assume that $1/\varepsilon$ is an integer. If $n \leq 1/\varepsilon$, then we can find the optimum value of (6)-(10) in constant time, so from now on we assume that $n > 1/\varepsilon$.

Following the method of Chekuri and Khanna, firstly we guess a value $\mathcal{O}$ between $(1-\varepsilon)\mathrm{OPT}^S_{\bar{x}^B}$ and $\mathrm{OPT}^S_{\bar{x}^B}$. Since we do not know the value of $\mathrm{OPT}^S_{\bar{x}^B}$, we define a set of numbers such that one of them will do. That is, the guesses will be numbers of the form $p^{S(\bar{x}^B)}_{\max}(1+\varepsilon)^i$ for some non-negative integer i. The set of guesses is $G = \{p^{S(\bar{x}^B)}_{\max}(1+\varepsilon)^i \mid 0 \leq i \leq g\}$, where g is a sufficiently large integer. To bound g, observe that $p^{S(\bar{x}^B)}_{\max} \leq \mathrm{OPT}^S_{\bar{x}^B} \leq n p^{S(\bar{x}^B)}_{\max}$ holds, and therefore, it is no use to guess numbers exceeding $n p^{S(\bar{x}^B)}_{\max}$. Now we can bound g as follows.

Proposition 1 $g \leq \lfloor 2\varepsilon^{-1} \ln n \rfloor$.

Proof We limit g by using the inequality $p^{S(\bar{x}^B)}_{\max}(1+\varepsilon)^g \leq n p^{S(\bar{x}^B)}_{\max}$. After simplification we get $(1+\varepsilon)^g \leq n$. Taking the logarithm of both sides yields $g\ln(1+\varepsilon) \leq \ln n$. Since $\ln(1+\varepsilon) \geq \varepsilon/2$ for $\varepsilon \leq 1$, we have $g\varepsilon/2 \leq \ln n$, which implies our claim. □

Clearly, we have a polynomial number of guesses in n, and one of them will satisfy $(1-\varepsilon)\mathrm{OPT}^S_{\bar{x}^B} \leq \mathcal{O} \leq \mathrm{OPT}^S_{\bar{x}^B}$. For each guess $\mathcal{O} \in G$, we will define a new problem instance of (6)-(10) obtained by dropping those small jobs with $p_j < \varepsilon\mathcal{O}/n$, and rounding down the processing time p_j of the remaining jobs to the nearest value p^*_j chosen from the set $P^* =$

$\{(\varepsilon\mathcal{O}/n)(1+\varepsilon)^{i-1} \mid 1 \le i \le h\}$, where h is the largest integer such that the rounded job processing times do not exceed $\mathcal{O}$, i.e.,

$$(\varepsilon\mathcal{O}/n)(1+\varepsilon)^{h-1} \le \mathcal{O}. \tag{11}$$

Proposition 2 $h \le \lfloor 4\varepsilon^{-1}\ln n \rfloor + 1$.

Proof To limit h from above, we rearrange (11) to obtain $(1+\varepsilon)^{h-1} \le n/\varepsilon$. Taking the logarithm of both sides yields $(h-1)\ln(1+\varepsilon) \le \ln(n/\varepsilon) \le 2\ln n$, where we used the assumption $n > 1/\varepsilon$. Since $\ln(1+\varepsilon) \ge \varepsilon/2$, we finally obtain $h - 1 \le 4\varepsilon^{-1}\ln n$, where the right-hand-side can be rounded down as h is integral. □

Subsequently we will show how to find an assignment x^S of small jobs to supply periods such that $\sum_{\ell=1}^{q-1}\sum_{j\in S(\bar{x}^B)} p_j^* x_{\ell,j}^S \ge (1-O(\varepsilon))\mathcal{O}$. To this end, we introduce job classes $S_1,\dots,S_h$, where S_i contains all the small jobs with rounded processing time $y_i = (\varepsilon\mathcal{O}/n)(1+\varepsilon)^{i-1}$. An optimal solution $\tilde{x}$ of (6)-(10) determines the subset of jobs from each S_i assigned to each supply period. For each $\ell = 1,\dots,q-1$ and $i = 1,\dots,h$, we will guess approximately the value of $y_i\sum_{j\in S_i}\tilde{x}_{\ell j}^S$ with $k_i^\ell(\varepsilon\mathcal{O}/h)$, where k_i^ℓ is a non-negative integer.

Proposition 3 *For a guessed objective value $\mathcal{O} \in G$, to approximate $y_i\sum_{j\in S_i}\tilde{x}_{\ell j}^S$, the largest possible k_i^ℓ value is at most h/ε.*

Proof Since we want to approximate the value of $y_i\sum_{j\in S_i}\tilde{x}_{\ell j}^S$, which is bounded by the guess $\mathcal{O} \in G$, $k_i^\ell(\varepsilon\mathcal{O}/h) \le \mathcal{O}$ implies $k_i^\ell \le h/\varepsilon$. □

We also have to specify which jobs from S_i to assign to supply period ℓ whose total rounded processing time is at least $k_i^\ell(\varepsilon\mathcal{O}/h)$. To this end, we apply the following procedure.

Algorithm Job picking:

1) Order the jobs in each S_i in non-decreasing a_j order.
2) For each S_i, $i = 1,\dots,h$, in turn do the following:
3) Choose the subset $U_i^1 \subseteq S_i$ of smallest $\sum_{j\in U_i^1} a_j$ value with $y_i|U_i^1| = \sum_{j\in U_i^1} p_j^* \ge k_i^1(\varepsilon\mathcal{O}/h)$. In general, U_i^ℓ is chosen from $S_i \setminus \left(\bigcup_{\kappa=1}^{\ell-1} U_i^\kappa\right)$ such that $\sum_{j\in U_i^\ell} a_j$ is minimal with $y_i|U_i^\ell| \ge k_i^\ell(\varepsilon\mathcal{O}/h)$, $\ell = 2,\dots,q-1$. If $k_i^\ell = 0$ for some $\ell \in \{1,\dots,q-1\}$, then $U_i^\ell = \emptyset$.
4) If we cannot pick enough elements for some k_i^ℓ from the (remaining) S_i, then the procedure fails, otherwise it outputs the sets U_i^ℓ.

The Job picking procedure is illustrated in Fig. 1, where the schedule of the big jobs is shown on the top, whereas the one obtained after inserting the small jobs is depicted in the bottom of the figure.

For $\ell = 1,\dots,q-1$, let $U^\ell(k_1^\ell,\dots,k_h^\ell) = \bigcup_{i=1}^h U_i^\ell$ be the set of jobs picked for the h tuple $(k_1^\ell,\dots,k_h^\ell)$. We define the *assignment of small jobs in the $q-1$ sets $U^\ell(k_1^\ell,\dots,k_h^\ell)$, $\ell = 1,\dots,q-1$, to supply periods* by a $(q-1)\times n$ binary vector x^U as follows:

$$x_{\ell,j}^U = \begin{cases} 1 \text{ if } j \in U^\ell(k_1^\ell,\dots,k_h^\ell), \\ 0 \text{ otherwise.} \end{cases} \qquad \ell = 1,\dots,q-1.$$

Lemma 6 *For any $\mathcal{O} \le \mathrm{OPT}_{\bar{x}^B}^S$, there exists an $h(q-1)$ tuple $(k_1^1,\dots,k_h^{q-1})$ such that the assignment x^U of small jobs to supply periods corresponding to the $q-1$ sets $U^\ell(k_1^\ell,\dots,k_h^\ell)$, $\ell = 1,\dots,q-1$, satisfies (7), and also the constraints*

$$\sum_{j\in\mathcal{S}} p_j^* x_{\ell,j} \le \max\{0, \tau_{\ell+1} - \bar{C}_\ell^B\} + \varepsilon p_{\mathrm{sum}},\ \ell < q \tag{12}$$

and

$$\sum_{\ell=1}^{q-1}\sum_{i=1}^{h} y_i \left(\sum_{j\in S_i} x_{\ell j}\right) \ge (1-(q+1)\varepsilon)\mathcal{O}.$$

(Constraint (12) is obtained from (8) by replacing p_j by p_j^*.)

Proof Take an optimal solution $\tilde{x}^S$ of (6)-(10) (for the original p_j values). Since $p_j \ge p_j^*$, $\tilde{x}^S$ satisfies (12) as well. Let the set $\tilde{U}_i^\ell$ consist of those small jobs J_j with $\tilde{x}_{\ell,j} = 1$ and $j \in S_i$, where we neglect those jobs with $p_j < \varepsilon\mathcal{O}/n$. Define $k_i^\ell = \lfloor p^*(\tilde{U}_i^\ell)h/(\varepsilon\mathcal{O})\rfloor$.

By applying the job picking procedure (described before this lemma), to the $h(q-1)$ tuple $(k_1^1,\dots,k_h^{q-1})$, the sets U_i^ℓ, may differ from the sets $\tilde{U}_i^\ell$. The reason is that there may exist distinct jobs J_j and J_k of the same rounded processing time such that $j \in \tilde{U}_i^\ell$, $k \in \tilde{U}_i^\kappa$ for some i, but $1 \le \ell < \kappa \le q-1$, and $a_j > a_k$, or some $\tilde{U}_i^\ell$ is not of smallest total weight with respect to the a_j. However, $\sum_{\ell=1}^t\sum_{j\in U_i^\ell} a_j \le \sum_{\ell=1}^t\sum_{j\in\tilde{U}_i^\ell} a_j$, and $|U_i^t| \le |\tilde{U}_i^t|$ for $t = 1,\dots,q-1$, and $i = 1,\dots,h$. Hence, the assignment x^U of small jobs to supply periods with respect to the $q-1$ sets $U^\ell(k_1^\ell,\dots,k_h^\ell) = \bigcup_{i=1}^h U_i^\ell$, $\ell = 1,\dots,q-1$, satisfies the constraints (7) and (12). Moreover, the value of the assignment is

$$\begin{aligned}
\sum_{\ell=1}^{q-1}\sum_{i=1}^{h} y_i \left(\sum_{j\in S_i} x_{\ell j}^U\right) &\ge \sum_{\ell=1}^{q-1}\sum_{i=1}^{h} k_i^\ell(\varepsilon\mathcal{O}/h) \\
&\ge \sum_{\ell=1}^{q-1}\sum_{i=1}^{h} p^*(\tilde{U}_i^\ell) - (q-1)\varepsilon\mathcal{O} \\
&\ge \sum_{\ell=1}^{q-1}\sum_{i=1}^{h} \frac{p(\tilde{U}_i^\ell)}{(1+\varepsilon)} - q\varepsilon\mathcal{O} \\
&\ge (1-(q+1)\varepsilon)\mathcal{O},
\end{aligned}$$

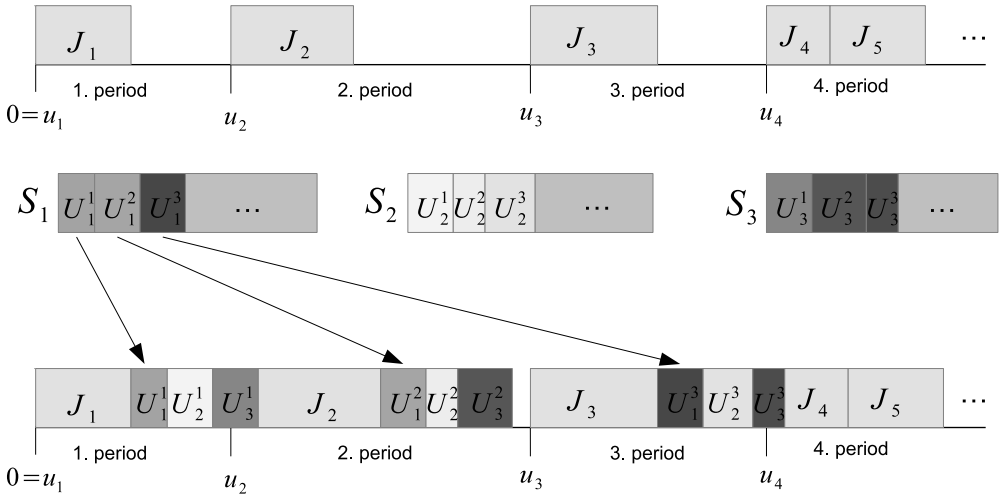

Fig. 1 Illustration of the Job picking procedure with $h = 3$.

where the first inequality follows from the definition of the sets $U^\ell(k_1^\ell, \ldots, k_h^\ell)$ and that of x^U, the second from the definition of the k_i^ℓ values, the third from the inequality $p_j^* \geq p_j/(1+\varepsilon) - \varepsilon\mathcal{O}/n$ (since those jobs with $p_j < \varepsilon\mathcal{O}/n$ are discarded, while for the remaining jobs we have $p_j^*(1+\varepsilon) \geq p_j$), and the last one is due to $\mathrm{OPT}^S_{\bar{x}^B} = \sum_{\ell=1}^{q-1}\sum_{i=1}^{h} p(\tilde{U}_i^\ell) \geq \mathcal{O}$, and $\mathcal{O}/(1+\varepsilon) \geq (1-\varepsilon)\mathcal{O}$ for $\varepsilon > 0$. □

Notice that the condition of $\mathcal{O} \leq \mathrm{OPT}^S_{\bar{x}^S}$ of Lemma 6 only excludes unattainable guesses for the optimum value of (6)-(10). We can limit the number of $h(q-1)$ tuples to be evaluated as follows.

Proposition 4 *The number of $h(q-1)$ tuples to be evaluated is $O(n^{O(\varepsilon^{1-q}+\varepsilon^{-2})})$. Evaluating a single tuple takes $O(qn)$ time. All the tuples can be evaluated in $O(h \cdot n \log n + (qn) \cdot n^{O(\varepsilon^{1-q}+\varepsilon^{-2})})$ time.*

Proof Recall that for a tuple $(k_1^1, \ldots, k_h^{q-1})$, the job picking procedure will produce sets U_i^ℓ such that $p^*(U_i^\ell) \geq k_i^\ell(\varepsilon\mathcal{O}/h)$, and since we want to approximate $\mathcal{O}$, it suffices to consider tuples with $\sum_{\ell=1}^{q-1}\sum_{i=1}^{h} k_i^\ell \leq h/\varepsilon$. A well known result in combinatorics says that the number of solutions of the inequality $x_1 + \cdots x_g \leq d$ among the nonnegative integers is $f = \binom{d+g}{g}$. Claim 2.4 of Chekuri and Khanna [7] says that if $d + g \leq \alpha g$ for some α, then $f = O(e^{\alpha g})$. Therefore, the number of those tuples $(k_1^1, \ldots, k_h^{q-1}) \in \mathbb{Z}_0^{h(q-1)}$ with $\sum_{\ell=1}^{q-1}\sum_{i=1}^{h} k_i^\ell \leq h/\varepsilon$ is $\binom{h/\varepsilon + h(q-1)}{h(q-1)}$, which is bounded by $O(n^{O(\varepsilon^{1-q}+\varepsilon^{-2})})$ (using $\alpha = 1 + 1/(\varepsilon(q-1))$), a polynomial of n for fixed ε and q.

To see the second part, notice that evaluating a single tuple $(k_1^1, \ldots, k_h^{q-1})$ consists of defining the vector x^U corresponding to the sets $U^\ell(k_1^\ell, \ldots, k_h^\ell)$, $\ell = 1, \ldots, q-1$, and then checking whether x^U satisfies (7) and (12). Notice that $S_1, \ldots, S_h$ need to be determined only once for each guess $\mathcal{O}$, and they can be sorted one-by-one in $O(\sum_{i=1}^{h} |S_i| \log_2 |S_i|)$ time in total, which can be very roughly bounded by $O(h \cdot n \log_2 n)$. After this pre-processing, computing x^U for a given $(k_1^1, \ldots, k_h^{q-1})$ boils down to determining the cardinality of the sets U_i^ℓ by simple divisions: $|U_i^\ell| = \lceil k_i^\ell(\varepsilon\mathcal{O}/h)/y_i \rceil$, since all jobs in S_i have the same rounded processing time y_i (see the Job picking procedure). Then we set the coordinates of x^U in $O(n)$ time in total by using the sorted sets S_i. Verifying the constraints (7) and (12) takes $O(qn)$ time. □

All in all, for each $\mathcal{O}$, we have a polynomial number of tuples to be evaluated. In order to find an ε-approximate solution of (6)-(10), we generate all the tuples with $\sum_{i=1}^{h}\sum_{\ell=1}^{q-1} k_i^\ell \leq h/\varepsilon$ in polynomial time, and check each tuple $(k_1^1, \ldots, k_h^{q-1})$ whether the assignment x^U of small jobs to supply periods corresponding to the set system $U^\ell(k_1^\ell, \ldots, k_h^\ell)$, $\ell = 1, \ldots, q-1$, satisfies (7) and (12). We choose the x^U giving an assignment $(\bar{x}^B, x^U)$ of smallest value $v((\bar{x}^B, x^U))$.

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

Proof Consider any $\mathcal{O} \in G$ with $(1-\varepsilon)\mathrm{OPT}(\bar{x}^B) \leq \mathcal{O} \leq \mathrm{OPT}(\bar{x}^B)$ (such an $\mathcal{O}$ exists by the definition of the set G). By Lemma 6, there is a $h(q-1)$ tuple $(k_1^1, \ldots, k_h^{q-1})$ such that $\sum_{\ell=1}^{q-1}\sum_{i=1}^{h} y_i \left(\sum_{j \in S_i} x_{\ell j}^U\right) = \sum_{\ell=1}^{q-1}\sum_{i=1}^{h} p^*(U_i^\ell) \geq (1-(q+1)\varepsilon)\mathcal{O}$, and x^U satisfies (7) and (12), where each set U_i^ℓ satisfies $k_i^\ell(\varepsilon\mathcal{O}/h) \leq p^*(U_i^\ell) < (k_i^\ell + 1)(\varepsilon\mathcal{O}/h)$, and x^U is the corresponding assignment of the small jobs to supply periods. We

may even assume that $\sum_{\ell=1}^{q-1}\sum_{i=1}^{h} k_i^\ell \le h/\varepsilon$, otherwise $\sum_{\ell=1}^{q-1}\sum_{i=1}^{h} p^*(U_i^\ell) \ge \sum_{\ell=1}^{q-1}\sum_{i=1}^{h} k_i^\ell(\varepsilon\mathcal{O}/h) > \mathcal{O}$ and we could decrease some of the k_i^ℓ values to meet $\sum_{\ell=1}^{q-1}\sum_{i=1}^{h} k_i^\ell \le h/\varepsilon$. We have

$$\begin{aligned}\sum_{\ell=1}^{q-1}\sum_{i=1}^{h}\left(\sum_{j\in S_i} p_j x_{\ell j}^U\right) &\ge \sum_{\ell=1}^{q-1}\sum_{i=1}^{h}\left(\sum_{j\in S_i} p_j^* x_{\ell j}^U\right)\\ &\ge (1-(q+1)\varepsilon)\mathcal{O}\\ &\ge (1-(q+1)\varepsilon)(1-\varepsilon)\mathrm{OPT}(\bar{x}^B)\\ &\ge (1-(q+2)\varepsilon)\mathrm{OPT}(\bar{x}^B),\end{aligned}$$

where the first inequality follows from $p_j \ge p_j^*$ for $j \in S(\bar{x}^B)$, the second from the choice of the tuple $(k_1^1,\ldots,k_h^{q-1})$, the third from the choice of $\mathcal{O}$, and the last from elementary calculations. Now since $p_j \le (1+\varepsilon)p_j^*$ if $p_j \ge \varepsilon\mathcal{O}/n$, and $p_j^* = 0$ otherwise, x^U violates (8) by at most $\varepsilon\sum_{j\in S(\bar{x}^B)} p_j^* \le \varepsilon p_{\mathrm{sum}}^S$ in total. Hence, x^U is an ε-approximate solution, and Lemma 5 implies that $(\bar{x}^B, x^U)$ is a solution of $IP(\bar{x}^B)$ of value at most $(1+O(\varepsilon))\mathrm{OPT}(\bar{x}^B)$.

Concerning the time complexity of the procedure, the number of big jobs is at most $1/\varepsilon$, since job j is big if and only if $p_j \ge \varepsilon p_{\mathrm{sum}}$. Hence, the number of assignments of big jobs to supply periods is at most $q^{1/\varepsilon}$, a constant. Therefore, the total running time is determined by the number of trials for $\mathcal{O}$ (for any fixed assignment of big jobs), and the complexity of generation and evaluation of all the tuples for each $\mathcal{O}$, which is $O(q^{1/\varepsilon}\cdot g\cdot(h\cdot n\log n + (qn)\cdot n^{O(\varepsilon^{1-q}+\varepsilon^{-2})}))$. Using Propositions 1 through 4, we get that the overall time complexity of the algorithm is $O(q^{1/\varepsilon}\cdot(2\varepsilon^{-1}\ln n)\cdot(n^2\varepsilon^{-1}\log^2 n + (qn)\cdot n^{O(\varepsilon^{1-q}+\varepsilon^{-2})}))$, a polynomial of n for fixed ε and q. □

6 A PTAS for $1|rm, ddc-agr, q=const|C_{\max}$

Finally, we sketch how to extend our PTAS to the more general $1|rm, ddc-agr, q=const|C_{\max}$ problem, when the resource requirements of jobs are agreeable. In fact, all we have to do is to order each set S_i in non-decreasing order for all the resource coordinates, and then we can apply a similar procedure as for the $1|rm=1, q=const|C_{\max}$ case. The only difference is that we have to deal with vectors of resource requirements and resource supplies rather than scalar quantities, but this increases the time complexity of the algorithm only by a factor of $|\mathcal{R}|$, which is polynomial in the input length.

7 Conclusions

In this paper we have devised exact and approximation algorithms for scheduling jobs on a single machine subject to resource constraints. This is just the first step, as there are several other objective functions for which no approximation schemes, or the hardness of approximation is known.

8 Acknowledgments

This work has been supported by the research grant "Digital, real-time enterprises and networks", OMFB-01638/2009. The research of Tamás Kis has been supported by the János Bólyai research grant BO/00412/12/3 of the Hungarian Academy of Sciences.

References

1. Blazewicz, J., Lenstra, J.K., Rinnooy Kan, A.H.G.: Scheduling subject to resource constraints: classification and complexity, *Discrete Applied Mathematics*, 5, 11-24 (1983)
2. Briskorn, D., Choi, B-C., Lee, K., Leung, J., Pinedo, M.: Complexity of single machine scheduling subject to nonnegative inventory constraints. *European Journal of Operational Research*, 207, 605–619, (2010)
3. Briskorn, D., Jaehn, F., Pesch, E.: Exact algorithms for inventory constrained scheduling on a single machine. *Journal of scheduling*, to appear (2012)
4. Carlier, J., Rinnooy Kan, A. H. G.: Scheduling subject to nonrenewable resource constraints. *Operations Research Letters*, 1, 52–55 (1982)
5. Carlier, J.: Problèmes d'ordonnancements à contraintes de ressources: algorithmes et complexité, Thèse d'état, 1984.
6. Carlier, J.: Scheduling under financial constraints, in R. Slowiński, J. Weglarz (eds), *Advances in Project Scheduling*, Elsevier, Amsterdam, pp. 187–224 (1989)
7. Chekuri, C., Khanna, S.: A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem, *SIAM J. Computing*, 35, 713–728 (2006)
8. Drótos, M., Kis, T.: Scheduling of inventory releasing jobs to minimize a regular objective function of delivery times, *Journal of Scheduling*, article in press (2012)
9. Gafarov, E.R, Lazarev, A.A., Werner, F.: Single machine scheduling problems with financial resource constraints: Some complexity results and properties, *Mathematical Social Sciences*, 62, 7–13 (2011)
10. Garey, M. R., Johnson, D. S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco (1979)
11. Graham, R.L., Lawler E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, 5, 287-326 (1979)
12. Grigoriev, A., Holthuijsen, M., van de Klundert, J.: Basic scheduling problems with raw material constraints, *Naval Research of Logistics*, 52, 527–553 (2005)

13. Neumann, K., Schwindt, C.: Project scheduling with inventory constraints. *Mathematical Methods of Operations Research*, 56, 513–533 (2002)
14. Slowinski, R.: Preemptive scheduling of independent jobs on parallel machines subject to financial constraints, *European J. Operational Research*, 15, 366–373 (1984)
15. Toker, A., Kondakci, S., Erkip, N.: Scheduling under a non-renewable resource constraint. *J. Operational Research Society*, 42, 811–814 (1991)