Egerváry Research Group
on Combinatorial Optimization

Technical reportS

TR-2019-5. Published by the Egerváry Research Group, Pázmány P. sétány 1/C, H–1117, Budapest, Hungary. Web site: www.cs.elte.hu/egres. ISSN 1587–4451.

New complexity and approximability results for minimizing the total weighted completion time on a single machine subject to non-renewable resource constraints

Péter Györgyi and Tamás Kis

March 21, 2019

New complexity and approximability results for minimizing the total weighted completion time on a single machine subject to non-renewable resource constraints

Péter Györgyi and Tamás Kis

Abstract

In this paper we consider a single machine scheduling problem with additional non-renewable resource constraints and the total weighted completion time objective. There are only a few complexity and approximability results for this problem. We consider special cases and describe new complexity results and also an FPTAS for a variant which is extended to instances with high-multiplicity jobs. We also prove some non-trivial approximation results for a simple greedy algorithm.

1 Introduction

Single machine scheduling is one of the oldest scheduling problems with many theoretical results and practical applications. In the recent years the importance of non-renewable resources (like raw materials, energy or money) is increasing. These resources are consumed by the jobs when the machine starts to process them. There is an initial stock, and some additional supplies arrive at given supply dates and in known quantities. Since the early 80s several papers examined the problem, but mainly considered "min max" type objectives, such as the makespan, or the maximum lateness. In this paper we focus on the total (weighted) completion time objective and prove some new complexity and approximability results.

Formally, we have a set of n jobs $\mathcal{J}$ to be scheduled on a single machine, and a non-renewable resource. Each job j has a processing time $p_j > 0$, a weight $w_j > 0$, and a resource requirement $a_j \geq 0$. We have q supplies from the resource at $0 = u_1 < u_2 < \ldots < u_q$ in quantities $\tilde{b}_\ell \geq 0$ for $\ell = 1, \ldots, q$, where the first supply can be considered as an initial stock. We can schedule a job j only if the inventory level is at least a_j at the start time S_j of j, and since the job consumes this amount of resource, the inventory level decreases by a_j. In other words, in a feasible schedule at each time point t the total supply until t is at least the total demand of those jobs starting not later than t, i.e., if $u_\ell \leq t$ is the last supply date before t, then $\sum_{j \in \mathcal{J}: S_j \leq t} a_j \leq \sum_{\ell'=1}^{\ell} \tilde{b}_{\ell'}$. We aim at finding a feasible schedule S that minimizes

the total weighted completion time $\sum_{j\in\mathcal{J}} w_j C_j$, where $C_j = S_j + p_j$. We denote our problem using the standard $\alpha|\beta|\gamma$ notation by $1|nr = 1|\sum w_j C_j$, where '$nr = 1$' indicates that there is only one type of non-renewable resource. Observe that we can assume that the total requirements of the jobs is equal to the total amount of supplied resources, hence at least one job starts not earlier than u_q.

We will consider some problems with high-multiplicity encoding, in which identical jobs are represented compactly by providing the common job parameters along with the number of jobs with these parameters. Hence, the length of the input depends only logarithmically on the number of jobs, see Section 1.3 for details. We use the notation '*hme*' in the β field for problems that use high-multiplicity encoding.

Scheduling with non-renewable resources is not only theoretically challenging, but it occurs frequently in practice. E.g., [10] examines the continuous casting stage of steel production in which hot metal is the non-renewable resource supplied by a blast furnace. A similar problem is studied in [3] at a shoe-firm and there are examples also in the consumer goods industry and in computer assembly [18]. Note that the problem is a special case of the resource constrained project scheduling problem, which has several further practical applications. We summarize the most important antecedents of this research in Section 1.1.

1.1 Previous work

The total weighted completion time objective in a single machine environment without additional resource constraints ($1||\sum w_j C_j$) is solvable in polynomial time, a classical result of Smith [17]. This objective function is studied in several papers, see e.g., [9], [1], or [13].

Non-renewable resource constraints in the context of machine scheduling has been introduced by Carlier [2], and by Slowinski [16]. For the total weighted completion time objective, Carlier proved that $1|nr = 1|\sum w_j C_j$ is strongly NP-hard. This result was repeated by Gafarov et al. [4], who also examined a variant where the supply dates are equidistant and each supplied amount $\tilde{b}_\ell$ is the same. In [12], it is proved that the problem is still NP-hard (in the weak sense) if there are only two supplies, and also an FPTAS is devised for this special case. In a recent paper, Györgyi and Kis [8] discuss some polynomially solvable special cases of $1|nr = 1|\sum w_j C_j$, and identify new NP-hardness variants, e.g., when $p_j = w_j = a_j$ for each job j. That paper also describes a 2-approximation algorithm for the above variant, and a PTAS when $p_j = w_j$, and the number of supplies (q) is a constant.

There are several papers for the makespan and the maximum lateness objective. For instance, in [19] it is proved that the single machine makespan minimization problem is equivalent to the two machine flowshop problem if the amount supplied at each time unit is the same, while in [7], the approximability of parallel machine scheduling under non-renewable resource constraints is investigated with the makespan and the maximum lateness objectives.

According to our best knowledge high-multiplicity scheduling problems were first examined by [14], and the term was coined by Hochbaum and Shamir [11]. We also refer to [6], where several high-multiplicity scheduling problems with non-renewable

resource constraints are examined, but only for the makespan and for the maximum lateness objectives.

1.2 Main results

Firstly, we investigate the complexity and approximability of the problem $1|nr = 1, a_j = \bar{a}, q = const. \,|\sum C_j$. For our complexity result, we need a high-multiplicity encoding of the input.

Theorem 1. *The problem* $1|nr = 1, a_j = 1, q = 2, hme \,|\sum C_j$ *is NP-hard.*

In fact, we reduce the NP-hard PARTITION problem to our scheduling problem, and we need a huge number of jobs in certain job classes.

For q constant, we have the following approximability result in standard encoding.

Theorem 2. *The problem* $1|nr = 1, a_j = \bar{a}, q = const \,|\sum C_j$ *admits an FPTAS.*

However, this FPTAS can be generalized to high-multiplicity encoding as well, provided the number of job classes, h, is fixed.

Theorem 3. *The problem* $1|nr = 1, a_j = \bar{a}, q = const, hme, h = const \,|\sum C_j$ *admits an FPTAS.*

The second problem studied in this paper is $1|nr = 1, p_j = 1, a_j = w_j \mid \sum w_j C_j$. When the number of supply dates is part of the input, we can show the following.

Theorem 4. *Scheduling the jobs in non-increasing* w_j *order is a 3-approximation algorithm for* $1|nr = 1, p_j = 1, w_j = a_j|\sum w_j C_j$.

However, for $q = 2$, more can be said:

Theorem 5. *Scheduling the jobs in non-increasing* w_j *order is a 2-approximation algorithm for* $1|nr = 1, p_j = 1, w_j = a_j, q = 2|\sum w_j C_j$.

We remark that this theorem remains valid in case of high-multiplicity encoding of the input.

The above results give partial answers to the open questions raised in the end of our previous paper [8].

1.3 Terminology

If there are many identical jobs, then the input can be described compactly using a *high-multiplicity encoding of the jobs*. Suppose the set of jobs can be partitioned into h classes such that all the jobs in the same class have the same parameters (processing time, job weight, and resource requirement). Then in the input there is a positive integer number h giving the number of job classes, and for each job class, we have a number s_i providing the number of identical jobs in the class, and 3 other numbers p_i, a_i and w_i specifying the common parameters of all the jobs in the class. If some of these values is the same over all the job classes, then it can be represented only once

in the input, but this further simplification does not decrease the size of the input significantly. The other input parameters are q, the number of supply periods, and the supply dates u_ℓ and supplied quantities $\tilde{b}_\ell$ for $\ell = 1, \ldots, q$.

A polynomial time algorithm on a high-multiplicity input must produce a compact schedule the size of which is bounded by a polynomial in the size of the high-multiplicity input. A natural schedule representation consists of $h{\cdot}q$ tuples $(\ell, i, t_{i\ell}, g_{i\ell})$, where ℓ is the index of the supply period, i is that of the job class, $t_{i\ell}$ is the start time of the first job from class $\mathcal{J}_i$ scheduled after u_ℓ, and $g_{i\ell}$ is the number of jobs from this class scheduled consecutively from $t_{i\ell}$ on. It is easy to see that one can check the feasibility, and compute the objective function value of such a schedule in polynomial time in the size of the input.

1.4 Notation

n	number of jobs
q	number of supply periods
j	job index
ℓ	index of supply period
p_j	processing time of job j
$\bar{a}$	resource requirement of any job
u_ℓ	the ℓ^{th} supply date
$\tilde{b}_\ell$	quantity supplied at u_ℓ
b_ℓ	total resource supply over the first ℓ supplies, i.e., $\sum_{k=1}^{\ell} \tilde{b}_k$
n_ℓ	total number of jobs that can be served from the first ℓ supplies

If all the jobs have the same resource requirement $\bar{a}$, we can determine in advance the total number of jobs that can be served from the first ℓ supplies. That is, $n_\ell = \lfloor b_\ell / \bar{a} \rfloor$. Since $n\bar{a} = \sum_{j=1}^{n} a_j = b_q$, we have $n_q = n$.

2 $1|nr = 1, a_j = 1, q = 2, hme| \sum C_j$ is NP-hard

In this section we prove Theorem 1. In that proof we will use the following lemma several times:

Lemma 1. *Let t be an arbitrary time point and S be an arbitrary feasible schedule of an instance of $1|nr = 1, hme| \sum C_j$. If there are k jobs with the same processing time p_j scheduled without idle time from time point t (in the time interval $[t, t + k \cdot p_j]$), then their contribution to the objective value of S is $kt + \binom{k+1}{2} \cdot p_j$.*

Proof. Let j' be the job in position k' ($1 \leq k' \leq k$) among these jobs. Since the completion time of j' is $t + k' \cdot p_j$, the contribution of these jobs to the objective is

$$\sum_{k'=1}^{k} (t + k' \cdot p_j) = kt + \left(\sum_{k'=1}^{k} k' \right) \cdot p_j = kt + \binom{k+1}{2} \cdot p_j.$$

□

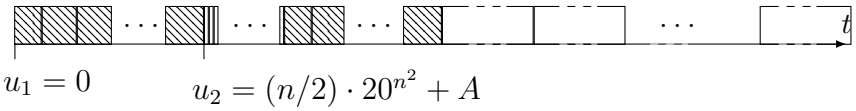

Figure 1: If the answer to I is 'yes' then the value of S is at most V'. The medium jobs are hatched.

Proof of Theorem 1. An instance of EQUAL-CARDINALITY-PARTITION is characterized by a positive even integer number n, and a set of n items with item sizes $e_1, \ldots, e_n \in \mathbb{Z}_{\geq 0}$ such that $\sum_i e_i = 2A$ for some integer A, and $e_i \leq 2^{n^2}$ (the last inequality follows from the proof of NP-hardness from [5]). Question: is there a subset S of items such that $|S| = n/2$ and $\sum_{i \in S} e_i = A$?

We reduce the EQUAL-CARDINALITY-PARTITION problem to our scheduling problem. Let I be an instance of the former problem, the corresponding instance I' of the scheduling problem is the following: we have $n' = 2 \cdot 200^{n^2} + n$ jobs and two supply dates, $u_1 = 0$ and $u_2 = (n/2) \cdot 20^{n^2} + A$ with $b_1 = n/2$ and $b_2 = 2 \cdot 200^{n^2} + n/2$, respectively. $J_1, J_2, \ldots, J_n$ are so called *medium* jobs with $p_j = 20^{n^2} + e_j$ ($j = 1, 2, \ldots, n$). We have 200^{n^2} small jobs with $p_j = 1$, and 200^{n^2} big jobs with $p_j = 200^{n^2}$. Let $\mathcal{J}_s, \mathcal{J}_m$ and $\mathcal{J}_b$ denote the set of small, medium and big jobs, respectively. The question is if there exists a feasible schedule of total job completion time at most $V' = V_s + V_m + V_b$, where

$$
\begin{aligned}
V_s &:= 200^{n^2} u_2 + \binom{200^{n^2} + 1}{2}, \\
V_m &:= 2 \cdot (20^{n^2} + A) \cdot \binom{n/2 + 1}{2} + (u_2 + 200^{n^2}) \cdot n/2, \\
V_b &:= 200^{n^2} \cdot (u_2 + 200^{n^2} + 20^{n^2} \cdot n/2 + A) + 200^{n^2} \cdot \binom{200^{n^2} + 1}{2} \\
&= 200^{n^2} \cdot \sum_{j \in \mathcal{J}_m \cup \mathcal{J}_s} p_j + 200^{n^2} \cdot \binom{200^{n^2} + 1}{2}.
\end{aligned}
$$

If the answer to I is 'yes', then consider the following schedule for I': schedule the medium jobs corresponding to the elements of S from 0 to u_2 in non-decreasing p_j order, then schedule the remaining jobs in non-decreasing p_j order from u_2, i.e., there are small jobs in the time interval $\left[u_2, u_2 + 200^{n^2}\right]$, medium jobs in the time interval $\left[u_2 + 200^{n^2}, u_2 + 200^{n^2} + A\right]$, and then there are the big jobs in the time interval $\left[u_2 + 200^{n^2} + 20^{n^2} \cdot n/2 + A, u_2 + 200^{n^2} + 20^{n^2} \cdot n/2 + A + 200^{2n^2}\right] = \left[\sum_{j \in \mathcal{J}_m \cup \mathcal{J}_s} p_j, \sum_{j \in \mathcal{J}_b \cup \mathcal{J}_m \cup \mathcal{J}_s} p_j\right]$ see Figure 1. Note that due to Lemma 1 the contribution of the small jobs to the objective function value is V_s, that of the medium jobs is at most V_m, while the contribution of the big jobs is V_b. Therefore, the total job completion time of this schedule is at most V'.

If the answer to I is 'no', then we claim that any feasible schedule has a larger

objective function value than V'. Let S^2 denote an arbitrary optimal schedule. We distinguish three cases.

- There is a big job that starts before u_2. Then necessarily all the other big jobs must be scheduled after all the small and medium jobs at the end of S^2. The contribution of the big jobs is at least $200^{n^2}+(200^{n^2}-1)\cdot(200^{n^2}+\sum_{j\in\mathcal{J}_s\cup\mathcal{J}_m} p_j)+ 200^{n^2}\cdot\binom{200^{n^2}}{2} \geq V_b - \sum_{j\in\mathcal{J}_m\cup\mathcal{J}_s} p_j$.

  There are at most $n/2-1$ other jobs that start before u_2 in S^2, thus there are at least $200^{n^2}+(n/2+1)$ small and medium jobs that start after the first big job, i.e., not earlier than 200^{n^2}. The contribution of these jobs to the optimum value is at least $(200^{n^2}+(n/2+1))\cdot 200^{n^2}+\binom{200^{n^2}+(n/2+1)+1}{2}$. This is clearly larger than $V_s+V_m+\sum_{j\in\mathcal{J}_m\cup\mathcal{J}_s} p_j$, thus the objective value of S^2 is larger than V'.

- If there are $k\geq 1$ small jobs that start before u_2, but each big job starts after u_2 in S^2, then at least $n/2+k$ medium jobs start after u_2 (since at most $n/2$ jobs may start before u_2 by the resource constraint), and the machine must be idle in $[u_2-k\cdot 20^{n^2}, u_2]$. This means that each big job starts not earlier than $\sum_{j\in\mathcal{J}_m\cup\mathcal{J}_s} p_j + k\cdot 20^{n^2}$ in S^2, because these jobs can start after all small and medium jobs in an optimal schedule. Hence, the contribution of the big jobs to the objective function value of S^2 is at least $V_b+k\cdot 200^{n^2}\cdot 20^{n^2}$.

  The contribution of the small and medium jobs is at least $\binom{k+1}{2}+(n/2-k)\cdot k+ \binom{n/2-k+1}{2}\cdot 20^{n^2}+(200^{n^2}-k)u_2+\binom{200^{n^2}-k+1}{2}+(n/2+k)\cdot(u_2+200^{n^2}-k)+\binom{n/2+k+1}{2}\cdot 20^{n^2}$, because in an optimal schedule the jobs are ordered in each period in a non-decreasing p_j order, i.e., both before and after u_2, the small jobs precede the medium jobs. Note that, $\binom{k+1}{2}+(200^{n^2}-k)u_2+\binom{200^{n^2}-k+1}{2} \geq V_s-k\cdot(u_2+200^{n^2})$, while $(n/2-k)\cdot k+\binom{n/2-k+1}{2}\cdot 20^{n^2}+(n/2+k)\cdot(u_2+200^{n^2}-k)+\binom{n/2+k+1}{2}\cdot 20^{n^2} \geq V_m - n^2A$. Therefore, the objective function value of S^2 is at least $V_b+V_m+V_s+k\cdot 200^{n^2}\cdot 20^{n^2}-n^2A-k\cdot(u_2+200^{n^2}) > V'$.

- If only medium jobs start before u_2, then the remaining jobs are scheduled after the maximum of u_2 and the completion of jobs scheduled before u_2, in non-decreasing processing time order. Since the answer to I is 'no', thus either there is a medium job that starts before u_2 and finishes after it, or there is idle time before u_2.

  In the former case, the big jobs are scheduled in $\left[\sum_{j\in\mathcal{J}_m\cup\mathcal{J}_s} p_j, \sum_{j\in\mathcal{J}_b\cup\mathcal{J}_m\cup\mathcal{J}_s} p_j\right]$, which means their contribution to the objective is V_b. Observe that, in this case exactly $n/2$ medium jobs start before u_2, because the total processing time of any $n-1$ medium jobs is less than u_2 and there can be at most $n/2$ jobs before u_2 due to the resource constraint of the first period. The contribution of the medium jobs is at least $2\cdot\binom{n/2+1}{2}\cdot 20^{n^2}+n/2\cdot(u_2+200^{n^2}) = V_m - 2A\cdot\binom{n/2+1}{2}$, because there are $n/2$ medium jobs that start not earlier than $u_2+200^{n^2}$ (after all small jobs are scheduled after u_2). The contribution of the small jobs is at

least $(200^{n^2}+1)u_2+\binom{200^{n^2}+1}{2} = V_s+200^{n^2}$, because these jobs are scheduled after the last medium job of the first period completes, i.e., not earlier than $200^{n^2}+1$. Since $200^{n^2} > 2A \cdot \binom{n/2+1}{2}$, the value of S_2 is larger than $V' = V_b + V_m + V_s$.

In the latter case, we can suppose that the number of the medium jobs start before u_2 is $n/2$, otherwise we would schedule a small job before u_2 to decrease the value of the schedule. Since the machine is idle in $[u_2 - 1, u_2]$, the big jobs cannot start before $\sum_{j\in\mathcal{J}_m\cup\mathcal{J}_s} p_j+1$, therefore their contribution to the objective is at least $V_b + 200^{n^2}$. The small jobs are scheduled in $\left[u_2, u_2 + 200^{n^2}\right]$, thus their contribution is exactly V_s. The contribution of the medium jobs is at least $2 \cdot (20^{n^2}) \cdot \binom{n/2+1}{2} + (u_2 + 200^{n^2}) \cdot n/2 = V_m - 2A \cdot \binom{n/2+1}{2}$, thus the objective function value of S^2 is at least $V_b + V_m + V_s + 200^{n^2} - 2A \cdot \binom{n/2+1}{2}$, which is clearly larger than $V' = V_b + V_m + V_s$.

□

3 FPTAS for $1|nr = 1, a_j = \bar{a}, q = const, hme, h = const| \sum C_j$

3.1 FPTAS for normal input

Firstly, we describe a dynamic program, and then we sketch how to turn it into an FPTAS. We assume that the jobs are indexed in non-increasing processing time order, i.e., $p_1 \geq \cdots \geq p_n$.

Our dynamic program is defined by an acyclic graph, where the nodes represent states, and the edges the transitions between them. Each state σ is a $3q$ tuple $(N_1^\sigma, \ldots, N_q^\sigma; P_1^\sigma, \ldots, P_q^\sigma; C_1^\sigma, \ldots, C_q^\sigma) \in \mathbb{R}^{3q}$, where N_ℓ^σ, P_ℓ^σ and C_ℓ^σ represent the total number, the total processing time, and the total completion time (if started at time 0) of those jobs assigned to the supply period ℓ. Note that if $j_1, \ldots, j_k$ are the jobs assigned to supply period ℓ in state σ, then $C_\ell^\sigma = kp_{j_1} + (k-1)p_{j_2} + \cdots + p_{j_k}$. The initial state is the all 0 vector. Consider any state $\sigma = (N_1^\sigma, \ldots, N_q^\sigma; P_1^\sigma, \ldots, P_q^\sigma; C_1^\sigma, \ldots, C_q^\sigma)$ with $\sum_{\ell'=1}^{\ell} N_{\ell'}^\sigma \leq n_\ell$ for each $\ell \in \{1, \ldots, q\}$, and with strict inequality for at least one ℓ. Let $j := \sum_{\ell=1}^{q} N_\ell^\sigma + 1$ the index of the next job to be scheduled. For each ℓ such that $\sum_{\ell'=1}^{\ell} N_{\ell'}^\sigma < n_\ell$, we define a successor state σ', unless it is already defined, as follows. In σ', the values of $N_\ell^{\sigma'}$, $P_\ell^{\sigma'}$, and $C_\ell^{\sigma'}$ are computed as $N_\ell^\sigma + 1$, $P_\ell^\sigma + p_j$, and $C_\ell^\sigma + (N_\ell^\sigma + 1) \cdot p_j$, respectively; while all other components are inherited from σ. The *terminal* states are those σ with $\sum_{\ell=1}^{q} N_\ell^\sigma = n$. The objective function value of a terminal state is computed as

$$\text{value}(\sigma) := \sum_{\ell=1}^{q} \left(C_\ell^\sigma + \max\left\{ u_\ell, \max_{\ell'<\ell} \left(u_{\ell'} + \sum_{k=\ell'}^{\ell-1} P_k^\sigma \right) \right\} \cdot N_\ell^\sigma \right). \tag{1}$$

Note that in the above expression, $C_\ell^\sigma + \max\left\{u_\ell, \max_{\ell'<\ell} u_{\ell'} + \sum_{k=\ell'}^{\ell-1} P_k^\sigma\right\} \cdot N_\ell^\sigma$ gives the total completion time of those jobs assigned to supply period ℓ. To see this, observe

that in σ, the first job in supply period ℓ starts at $\max\left\{u_\ell, \max_{\ell'<\ell}\left(u_{\ell'} + \sum_{k=\ell'}^{\ell-1} P_k^\sigma\right)\right\}$. Since there are N_ℓ^σ jobs assigned to supply period ℓ, we get the formula (1).

We determine all the final states, and choose the best one, i.e., with the smallest value. Note that resource feasibility is ensured by the definition of n_ℓ and the fact that $\sum_{k=1}^{\ell} N_k^\sigma \le n_\ell$ in each (terminal) state σ.

We claim that the running time of this procedure is pseudo polynomial. To see this, notice that any number in any state can by bounded by $n\,\mathrm{MAXNUM}$, where MAXNUM is the maximum number in the input. Since q is a constant, the number of states can be bounded by $\mathrm{SOL}(n,q)\cdot(n\,\mathrm{MAXNUM})^{3q}$, where $\mathrm{SOL}(n,q)$ is the total number of solutions of the Diophantine equation system $\sum_{k=1}^{\ell} N_k \le n_\ell$, $\ell = 1,\ldots,q$. This can be bounded by n^q. Therefore, the running time can be bounded by a polynomial in n and MAXNUM, if q is a constant. Therefore, we have proved the following:

Lemma 2. *The problem* $1|nr = 1, q = const, a_j = \bar{a}|\sum C_j$ *can be solved in pseudo-polynomial time.*

Luckily, this pseudo-polynomial time algorithm can be turned into an FPTAS under the same conditions.

Let $\Delta = 1 + \varepsilon/(2n)$. We shall use the following rounding function:

$$r(v) = \begin{cases} 0, & \text{if } v = 0 \\ \Delta^{\lceil \log_\Delta v \rceil}, & \text{if } v > 0. \end{cases}$$

A notable property of this function is that if $v_1,\ldots,v_t$ is a sequence of $t \le n$ non-negative numbers, and $g_i = r(v_i + g_{i-1})$, where $g_0 = 0$, then

$$\sum_{j=1}^{i} v_j \le g_i \le (1+\varepsilon)\sum_{j=1}^{i} v_j, \quad i = 1,\ldots,t.$$

The first inequality follows from $g(v) \ge v$, and the second from $(1+\alpha/n)^n \le e^\alpha \le 1 + 2\alpha$ for $0 \le \alpha < 1$, see [15].

We modify the dynamic program as follows. We consider rounded states $\tilde{\sigma}$, the initial state being the 3q-dimensional 0 vector. When computing a new state $\tilde{\sigma}'$ from a rounded state $\tilde{\sigma}$ by modifying the components $N_\ell^{\tilde{\sigma}}$, $\tilde{P}_\ell^{\tilde{\sigma}}$, and $\tilde{C}_\ell^{\tilde{\sigma}}$ (see above), we use the rounded values $r(\tilde{P}_\ell^{\tilde{\sigma}} + p_j)$, and $r(\tilde{C}_\ell^{\tilde{\sigma}} + (N_\ell^{\tilde{\sigma}} + 1)p_j)$ when determining the components $\tilde{P}_\ell^{\tilde{\sigma}'}$ and $\tilde{C}_\ell^{\tilde{\sigma}'}$, respectively.

For a terminal rounded state $\tilde{\sigma}$, let $r^{-1}(\tilde{\sigma})$ the set of those terminal states of the original dynamic program, that would yield $\tilde{\sigma}$ after iteratively rounding all of their P_ℓ and C_ℓ components obtained as sums of at most n numbers. Let $\sigma \in r^{-1}(\tilde{\sigma})$ be arbitrary. Now we can bound the value of a terminal state $\tilde{\sigma}$ as follows:

$$\mathrm{value}(\sigma) \le \mathrm{value}(\tilde{\sigma}) \le (1+\varepsilon)\sum_{\ell=1}^{q}\left(C_\ell^\sigma + \max\left\{u_\ell, \max_{\ell'<\ell}\left(u_{\ell'} + \sum_{k=\ell'}^{\ell-1} P_k^\sigma\right)\right\}\cdot N_\ell^\sigma\right).$$

But then, if σ^* is an optimal terminal state of the dynamic program without rounding, the algorithm with rounding will obtain some terminal $\tilde{\sigma}$ such that $\sigma^* \in r^{-1}(\tilde{\sigma})$, and thus, any terminal solution $\sigma \in r^{-1}(\tilde{\sigma})$ is of value at most $(1+\varepsilon)\,\text{value}(\sigma^*)$.

It remains to verify the time complexity of the dynamic program with the rounded states. The crucial factor in determining the running time is the number of distinct values of the components $\tilde{P}_\ell$ and $\tilde{C}_\ell$ of the rounded states. Since $\log_{\bar{\Delta}} \sum p_j = \ln \sum p_j / \ln \Delta \leq 4n \ln \sum p_j / \varepsilon$ for any $0 < \varepsilon < 1$, the number of distinct $\tilde{P}_\ell$ values is bounded by a polynomial $poly(|I|, 1/\varepsilon)$ in the size of the input and in $1/\varepsilon$. Since $C_\ell \leq n \sum p_j$, a similar statement is true for the rounded $\tilde{C}_\ell$ values. Therefore, the time complexity of the algorithm on any input I with n jobs is $O(n^q poly(|I|, 1/\varepsilon)^{2q})$. Hence, we proved Theorem 2.

3.2 FPTAS for *hme*-input

In this section we describe how to modify the FPTAS of the previous section to deal with *hme*-input. Recall that in such an input, there are h job classes and the number of jobs in class $\mathcal{J}_i$ is s_i. Since $\sum_{i=1}^{h} s_i$ may not be polynomially bounded in the size of the *hme*-input, the FPTAS of the previous section would have a pseudo-polynomial time complexity if applied directly to *hme*-input.

Firstly, we need a slightly different rounding function. Let $\bar{\Delta} := (1 + \varepsilon/(2h))$, and

$$\bar{r}(v) := \begin{cases} 0, & \text{if } v = 0, \\ \bar{\Delta}^{\lceil \log_{\bar{\Delta}} v \rceil}, & \text{if } v > 0. \end{cases}$$

Recall that the FPTAS of the previous section runs in n-stages (n is the number of the jobs), and in stage j, job j is assigned to one of the ℓ supply periods. We modify this strategy as follows. In the FPTAS for *hme*-input, there are h stages, one for each job class. We assume that $p_1 \geq \cdots \geq p_h$. The states of the new dynamic program are labeled by $3q+1$ tuples of the form $(i; N_1^\sigma, \ldots, N_q^\sigma; \tilde{P}_1^\sigma, \ldots, \tilde{P}_q^\sigma; \tilde{C}_1^\sigma, \ldots, \tilde{C}_q^\sigma)$, and they differ from the states of the dynamic program of the previous section in one important aspect: the first component is the index of the job class scheduled last. The initial state is the all-zero vector. Each arc connects a state σ at some stage $i-1 \geq 0$ with a state σ' at stage $i \leq h$, and it is labeled with a tuple $(\delta_{i1}, \ldots, \delta_{iq})$, where $\sum_{\ell=1}^{q} \delta_{i\ell} = s_i$, which provides the number of jobs from class $\mathcal{J}_i$ assigned to each of the supply periods. Since the number of such tuples is in the order of $\Omega(s_i^q)$, which is not bounded by a polynomial in the size of the *hme*-input in general, we cannot enumerate all the possible assignments of the jobs from class $\mathcal{J}_i$ to supply periods in an algorithm of polynomial running time in the size of the *hme*-input and in $1/\varepsilon$. For this reason, consider the quantities $(1+\varepsilon)^k$, for $k \in \mathcal{K}_i := \{0, \ldots, \lceil \log_{(1+\varepsilon)} s_i \rceil\}$. We will use the tuples $(k_1, \ldots, k_q)$, where each $k_\ell \in \mathcal{K}_i$. Let E_i be the set of eligible tuples, where a tuple $(k_1, \ldots, k_q)$ is *eligible* if and only if $\sum_{\ell=1}^{q} \lceil (1+\varepsilon)^{k_\ell} \rceil \geq s_i$. The number of eligible tuples is bounded by $|\mathcal{K}_i|^q$, which is bounded by $((2 \ln s_i)/\varepsilon)^q$, since $|\mathcal{K}_i| \leq (2 \ln s_i)/\varepsilon$ as a standard computation shows. However, this is polynomially bounded in the size of the *hme*-input and in $1/\varepsilon$. For each tuple $(k_1, \ldots, k_q) \in E_i$, we compute an assignment of jobs to supply periods by the following Allocation algorithm:

1. Let $t = s_i$, and $\ell = q$.
2. While $\ell > 0$ do
3. Let $\delta_{i\ell} := \min\{t, \lceil (1+\varepsilon)^{k_\ell} \rceil\}$, and $t := t - \delta_{i\ell}$
4. Let $\ell := \ell - 1$
5. End do
6. Output: $(\delta_{i1}, \ldots, \delta_{iq})$.

Observe that the jobs of class $\mathcal{J}_i$ are assigned backward, from supply period q to supply period 1. The use of this allocation strategy is in the proof of feasibility of the set of tuples corresponding to the optimal solution, as we will see later. For each distinct tuple $(\delta_{i1}, \ldots, \delta_{iq})$, a subsequent state σ' of σ is defined such that $\sigma' = (i; N_1^{\sigma'}, \ldots, N_q^{\sigma'}; \tilde{P}_1^{\sigma'}, \ldots, \tilde{P}_q^{\sigma'}; \tilde{C}_1^{\sigma'}, \ldots, \tilde{C}_q^{\sigma'})$, where $N_\ell^{\sigma'} := N_\ell^\sigma + \delta_{i\ell}$, $\tilde{P}_\ell^{\sigma'} := \bar{r}(\tilde{P}_\ell^\sigma + \delta_{i\ell} \cdot p_i)$, and $\tilde{C}_\ell^{\sigma'} := \bar{r}(\tilde{C}_\ell^\sigma + (\delta_{i\ell} N_\ell^\sigma + \delta_{i\ell} \cdot (\delta_{i\ell}+1)/2) \cdot p_i)$. If σ' is already stored at stage i, then the processing of σ' is finished. Otherwise, we check the *feasibility* of σ' by verifying the condition

$$\sum_{\ell'=1}^{\ell} N_{\ell'}^{\sigma'} \leq n_\ell, \text{ for } \ell = 1, \ldots, q-1. \tag{2}$$

We store σ' at stage i only if it satisfies (2).

The states obtained at stage h are the *terminal states*. Clearly, all terminal states represent feasible allocation of jobs to supply periods. Among the terminal states, we pick the one with smallest value, computed by the formula (1). The solution is obtained by repeatedly moving to the predecessor states until the initial state is reached. The arcs visited provide an eligible assignment from each E_i, that together determine a solution of the problem.

It remains to verify the approximation ratio and the time complexity of the algorithm. Consider an optimal solution of the scheduling problem, and for each job class $\mathcal{J}_i$ and supply period ℓ, let $n_{i\ell}^*$ denote the number of jobs from class $\mathcal{J}_i$ started in the interval $[u_\ell, u_{\ell+1})$, if $\ell < q$, and not before u_q if $\ell = q$. Let $k_{i\ell}$ be the smallest integer such that $n_{i\ell}^* \leq \lceil (1+\varepsilon)^{k_{i\ell}} \rceil$. Clearly, all the tuples $(k_{i1}, \ldots, k_{iq})$ are eligible. Let $(\delta_{i1}, \ldots, \delta_{iq})$ be the job allocations obtained from the $(k_{i1}, \ldots, k_{iq})$, and consider the sequence of states $\sigma_1, \sigma_2, \ldots, \sigma_h$ obtained by applying the job allocations $(\delta_{i1}, \ldots, \delta_{iq})$ in increasing order of the index i.

Claim 1. *The states $\sigma_1, \ldots, \sigma_h$ satisfy the condition (2), and for each i, σ_i is a state stored at stage i of the algorithm.*

Proof. Clearly, the algorithm will generate σ_1. If it satisfies the condition (2), then it will generate σ_2 from it, etc. So we have to prove that σ_h satisfies the condition (2), because it implies that all previous states do. By the rules of the Allocation algorithm,

$\sum_{\ell'=\ell}^{q} \delta_{i\ell'} \geq \sum_{\ell'=\ell}^{q} n^*_{i\ell'}$ for each ℓ and i. Consequently, $\sum_{\ell'=1}^{\ell} \delta_{i\ell'} \leq \sum_{\ell'=1}^{\ell} n^*_{i\ell'}$, since $s_i = \sum_{\ell=1}^{q} \delta_{i\ell} = \sum_{\ell=1}^{q} n^*_{i\ell}$. Since the optimal solution is feasible, we have

$$\sum_{\ell'=1}^{\ell} N_{\ell'}^{\sigma_h} = \sum_{i=1}^{h} \sum_{\ell=1}^{\ell} \delta_{i\ell'} \leq \sum_{i=1}^{h} \sum_{\ell'=1}^{\ell} n^*_{i\ell'} \leq n_\ell, \text{ for } \ell = 1, \ldots, q-1,$$

which proves our claim. □

Let σ be the state that is obtained from the initial state by applying the job allocations $(\delta_{i1}, \ldots, \delta_{iq})$ in increasing order of the index i, but without rounding the components P_ℓ and C_ℓ by $\bar{r}(\cdot)$. Using σ, the value of σ_h can be bounded as follows:

$$\text{value}(\sigma) \leq \text{value}(\sigma_h) < (1+\varepsilon) \sum_{\ell=1}^{q} \left(C_\ell^\sigma + \max \left\{ u_\ell, \max_{\ell' < \ell} \left(u_{\ell'} + \sum_{k=\ell'}^{\ell-1} P_k^\sigma \right) \right\} \cdot N_\ell^\sigma \right),$$

where the first inequality follows from the fact that σ_h is derived similarly to σ by applying the job allocations $(\delta_{i1}, \ldots, \delta_{iq})$ to the initial state, but the components P_ℓ and C_ℓ are rounded by $\bar{r}(\cdot)$ in each step. The second inequality follows from the properties of the rounding function. We have to relate the right-hand-side of the above expression to the value of the optimal solution. Let N_ℓ^*, P_ℓ^*, and C_ℓ^* denote the total number of jobs, the total processing time, and the total completion time (if started at time 0) of those jobs assigned to supply period ℓ in the optimal solution. Notice that $\lceil (1+\varepsilon)^{k_{i\ell}} \rceil \leq (1+\varepsilon) n^*_{i\ell}$. It follows that

$$\begin{aligned}
N_\ell^\sigma &\leq (1+\varepsilon) \sum_{i=1}^{h} n^*_{i\ell} = (1+\varepsilon) N_\ell^*, \\
P_\ell^\sigma &\leq (1+\varepsilon) \sum_{i=1}^{h} n^*_{i\ell} p_i = (1+\varepsilon) P_\ell^*, \text{ and} \\
C_\ell^\sigma &\leq (1+\varepsilon)^2 \sum_{i=1}^{h} \left(\left(\sum_{j=1}^{i} n^*_{i\ell} \cdot n^*_{j\ell} \right) + n^*_{i\ell}(n^*_{i\ell}+1)/2 \right) \cdot p_i = (1+\varepsilon)^2 C_\ell^*.
\end{aligned}$$

Clearly, the optimum value can be expressed as

$$OPT := \sum_{\ell=1}^{q} \left(C_\ell^* + \max \left\{ u_\ell, \max_{\ell' < \ell} \left(u_{\ell'} + \sum_{k=\ell'}^{\ell-1} P_k^* \right) \right\} \cdot N_\ell^* \right). \tag{3}$$

Therefore, the value of σ_h is at most $(1+\varepsilon)^3$ times the value of the optimal solution. Since the algorithm chooses the terminal state with smallest value, and σ_h is one of the terminal states, the best terminal state is at most $(1+\varepsilon)^3$ times away from the optimum.

Finally, the time complexity of the algorithm is proportional to the number of distinct $N_1, \ldots, N_q$ values that can be obtained by choosing an eligible tuple from each E_i. This can be bounded by $O(\Pi_{i=1}^{h} (2(\ln s_i)/\varepsilon)^q)$, which is bounded by $O((2(\ln \sum_{i=1}^{h} s_i)/\varepsilon)^{q \cdot h})$, a polynomial in the size of the *hme*-input and in $1/\varepsilon$, provided that q and h are fixed. Therefore, Theorem 3 is proved.

4 Approximation of $1|nr = 1, p_j = 1, w_j = a_j| \sum w_j C_j$

In this section first we prove Theorem 4, and then Theorem 5. For the sake of simpler notation, we assume that the jobs are indexed in non-increasing w_j order, i.e., $w_1 \geq w_2 \geq \ldots \geq w_n$.

Proof of Theorem 4. Let S be the solution obtained by scheduling the jobs in non-increasing w_j order, and S^* an optimal schedule. Let W_ℓ and W_ℓ^* be the total weight of the jobs that start in $[u_\ell, u_{\ell+1})$ in S, and in S^*, respectively. Let k_ℓ be the index of the last job that start before $u_{\ell+1}$ in S. Let G_ℓ and G_ℓ^* denote the length of the idle period in $[u_\ell, u_{\ell+1})$ in S and in S^*, respectively. Let $s_\ell := G_\ell - G_\ell^*$.

Note that $\sum_{\ell'=1}^{\ell} W_{\ell'} + w_{k_\ell+1} > \sum_{\ell'=1}^{\ell} W_{\ell'}^*$, thus $\sum_{\ell'=\ell+1}^{q} W_{\ell'} < \sum_{\ell'=\ell+1}^{q} W_{\ell'}^* + w_{k_\ell+1}$.

Note that both of G_ℓ and G_ℓ^* are at most $u_{\ell+1} - u_\ell$.

Since the jobs are scheduled in non-increasing w_j order in schedule S, the objective function value of this schedule is:

$$\sum_{j=1}^{n} j w_j + \sum_{\ell=1}^{q-1} G_\ell \left(\sum_{\ell'=\ell+1}^{q} W_{\ell'} \right) \leq \sum_{j=1}^{n} j w_j + \sum_{\ell=1}^{q-1} G_\ell \left(\sum_{\ell'=\ell+1}^{q} W_{\ell'}^* + w_{k_\ell+1} \right) \tag{4}$$

On the other hand, the optimum is at least

$$\sum_{j=1}^{n} j w_j + \sum_{\ell=1}^{q-1} G_\ell^* \left(\sum_{\ell'=\ell+1}^{q} W_{\ell'}^* \right),$$

thus the difference between the two values is at most

$$\sum_{\ell=1}^{q-1} s_\ell \left(\sum_{\ell'=\ell+1}^{q} W_{\ell'}^* \right) + \sum_{\ell=1}^{q-1} w_{k_\ell+1} G_\ell. \tag{5}$$

The first part of the above expression is obviously at most the optimum value (because $s_\ell \leq u_{\ell+1}$), while the second part can be bounded by $\sum_{\ell=1}^{q-1} w_{k_\ell+1}(u_{\ell+1} - u_\ell)$.

Since in S the jobs are scheduled in non-increasing w_j order, there is a job $j' \leq k_1+1$ that starts after u_2 in S^*. Suppose that it starts in $[u_{\ell_1}, u_{\ell_1+1})$. It contributes to the optimum by at least $w_{k_1+1} u_{\ell_1}$, which is at least $\sum_{\ell=1}^{\ell_1-1} w_{k_\ell+1}(u_{\ell+1} - u_\ell)$. Note that there is a job $j'' \leq k_{\ell_1} + 1$ that starts after u_{ℓ_1} in S^*. Suppose that it starts in $[u_{\ell_2}, u_{\ell_2+1})$, thus it contributes to the optimum by at least $w_{k_{\ell_1}+1} u_{\ell_2} \geq \sum_{\ell=\ell_1}^{\ell_2-1} w_{k_\ell+1}(u_{\ell+1} - u_\ell)$, etc.

Thus the second part of (5) is also at most the optimum, thus the total difference is at most two times the optimum, therefore S is a 3-approximation.

□

Proof of Theorem 5. Let S be the schedule found by the algorithm and S^* an optimal schedule. We use the same notation as in the proof of Theorem 4, but for simplification we introduce $G^* := G_1^*$. Note that if $G^* = 0$, then the algorithm yields an optimal schedule. Suppose for contradiction that there is an instance where the theorem is not true. Consider a counterexample I with minimal number of jobs. If J_1 is scheduled

Figure 2: Schedule S, where the jobs are in non-increasing w_j order.

from 0 in S^* (i.e., $S_1^* = 0$), then consider the instance I' obtained from I by dropping J_1 and by decreasing b_1 by w_1 and u_2 by 1. Then the algorithm gives a schedule S' such that $S'_j = S_j - 1$ for each $j = 2, \ldots, n$. Furthermore, the objective function value of S' is related to that of S as follows:

$$\sum_{j=2}^{n} w_j C'_j = \sum_{j=1}^{n} w_j (C_j - 1) = \sum_{j=1}^{n} w_j C_j - \sum_{j=1}^{n} w_j. \tag{6}$$

On the other hand, we can derive a new feasible schedule for I' from S^*: Let $\tilde{S}_j = S_j^* - 1$ for $j = 2, \ldots, n$. This schedule is again feasible, and its value is

$$\sum_{j=2}^{n} w_j \tilde{C}_j = \sum_{j=1}^{n} w_j (C_j^* - 1) = \sum_{j=1}^{n} w_j C_j^* - \sum_{j=1}^{n} w_j. \tag{7}$$

Comparing (6), and (7), we get that I' is also a counterexample with fewer jobs, which is a contradiction. From now on we assume that S_1^* is u_2.

Let J_k be the last job scheduled before u_2 in S, see Figure 2. We can describe the objective function value of S as a special case of (4), but now we choose a slightly different description due to technical reasons:

$$\sum_{j=1}^{n} w_j C_j = \sum_{j=1}^{k} j w_j + u_2 \sum_{j=k+1}^{n} w_j + \sum_{j=k+1}^{n} (j-k) w_j = \sum_{j=1}^{k} j w_j + (u_2 - k + 1) \sum_{j=k+1}^{n} w_j + \sum_{j=k+1}^{n} (j-1) w_j. \tag{8}$$

To determine a lower bound on the optimum of the problem, we determine the contribution of J_1 (recall that $S_1^* = u_2$), give a lower bound on the contribution of the other jobs:

$$\sum_{j=1}^{n} w_j C_j^* \geq (u_2 + 1) w_1 + \sum_{j=2}^{n} (j-1) w_j + (W_2^* - w_1) \cdot (G^* + 1). \tag{9}$$

The above expression is the value of the (not necessarily feasible) schedule, where J_1 starts at u_2 and the machine is idle in $[u_2 - G^*, u_2)$, which is an obvious lower bound on the optimum.

The difference among (8) and (9) is

$$\sum_{j=1}^{k} jw_j + (u_2 - k + 1) \sum_{j=k+1}^{n} w_j - (u_2 + 1)w_1 - \sum_{j=2}^{k}(j-1)w_j - (W_2^* - w_1)G^* -$$
$$W_2^* + w_1 = \sum_{j=1}^{k} w_j + (u_2 - k + 1)(W_1^* + W_2^* - \sum_{j=1}^{k} w_j) - u_2 w_1 - (W_2^* - w_1)G^* - W_2^*. \tag{10}$$

We need to prove that (10) cannot be larger than the optimum. Since $G^* \neq 0$ and $w_j = a_j$ for all job j, $\sum_{j=1}^{k+1} w_j > b_1$ follows, because otherwise we could have scheduled job $k+1$ earlier. However, $W_1^* \leq b_1$ because S^* is feasible, thus we have $W_1^* < \sum_{j=1}^{k+1} w_j$ and therefore the difference is at most

$$\sum_{j=1}^{k} w_j + (u_2 - k + 1)(W_2^* + w_{k+1}) - u_2 w_1 - (W_2^* - w_1)G^* - W_2^* =$$
$$\sum_{j=1}^{k} w_j + u_2 W_2^* + u_2(w_{k+1} - w_1) - kW_2^* - (k-1)w_{k+1} - (W_2^* - w_1)G^*. \tag{11}$$

Now, if $k = 0$, then then (11) simplifies to

$$u_2 W_2^* + w_1 - (W_2^* - w_1)G^*.$$

However, this last expression is a lower bound on the optimum value, since the contribution of those jobs that start after u_2 in S^* is at least $(u_2+1)W_2^*$ and largest-weight job starts at u_2 in S^* as well.

Finally, suppose that $k \geq 1$. Then (11) can be bounded from above by

$$\sum_{j=1}^{k} w_j + u_2 W_2^*,$$

because $W_2^* \geq w_1 \geq w_{k+1}$. Furthermore, $\sum_{j=1}^{n} w_j C_j^* \geq \sum_{j=1}^{k} w_j + u_2 W_2^*$, because each $C_j^* \geq 1$ and there are jobs with a total weight of at least W_2^* with a completion time of at least $u_2 + 1$ in S^*, thus the theorem follows.

□

5 Conclusion

We have shown several approximation results for different variants of $1|nr = 1|\sum w_j C_j$. However, there are still a lot of open problems in this topic. For instance, it is unknown whether there is a polynomial time constant factor approximation algorithm for $1|nr = 1|\sum w_j C_j$ or not. We have conjectured that scheduling the jobs in non-increasing w_j order is a factor 2 approximation algorithm for $1|nr = 1, p_j = 1, w_j = a_j|\sum w_j C_j$, but until now we could not prove it.

Acknowledgment

This work has been supported by the National Research, Development and Innovation Office – NKFIH, grant no. SNN 129178, and ED_18-2-2018-0006.

References

[1] A. Bachman, T. Cheng, A. Janiak, and C. Ng. Scheduling start time dependent jobs to minimize the total weighted completion time. *Journal of the Operational Research Society*, 53(6):688–693, 2002.

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

[3] S. Carrera, W. Ramdane-Cherif, and M.-C. Portmann. 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*, page 6. IFAC Publisher, 2010.

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

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

[6] A. Grigoriev. *High multiplicity scheduling problems.* PhD thesis, Maastricht University, 2003.

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

[8] P. Györgyi and T. Kis. Minimizing total weighted completion time on a single machine subject to non-renewable resource constraints. *Journal of Scheduling*, in press, 2019.

[9] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of operations research*, 22(3):513–544, 1997.

[10] O. Herr and A. Goel. Minimising total tardiness for a single machine scheduling problem with family setups and resource constraints. *European Journal of Operational Research*, 248:123–135, 2016.

[11] D. S. Hochbaum and R. Shamir. Strongly polynomial algorithms for the high multiplicity scheduling problem. *Operations Research*, 39(4):648–653, 1991.

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

[13] G.-S. Liu, J.-J. Li, H.-D. Yang, and G. Q. Huang. Approximate and branch-and-bound algorithms for the parallel machine scheduling problem with a single server. *Journal of the Operational Research Society*, pages 1–17, 2019.

[14] H. N. Psaraftis. A dynamic programming approach for sequencing groups of identical jobs. *Operations Research*, 28(6):1347–1359, 1980.

[15] P. Schuurman and G. J. Woeginger. Approximation schemes–a tutorial. *Unpublished manuscript*, 2004.

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

[17] W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics (NRL)*, 3(1-2):59–66, 1956.

[18] H. Stadtler and C. Kilger. *Supply Chain Management and Advanced Planning. Concepts, Models, Software, and Case Studies*. Springer, 4th edition, 2008.

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