Approximability of total weighted completion time with resource consuming jobs

Tamás Kis

September 8, 2015

Abstract

In this paper we study an extension of the single machine scheduling problem with the total weighted completion time objective, where there is a single non-renewable resource consumed by the jobs, having an initial stock and some additional replenishments over time. We prove that this problem is NP-hard in the strong sense, and provide an FPTAS for a special case with two supply dates.

Keywords Scheduling, approximation algorithms, non-renewable resource

1 Introduction

We will study single machine scheduling problems with resource consuming jobs and the total weighted completion time objective. In these problems, beside the machine and the jobs, there is a non-renewable resource (like raw material, energy, or money), consumed by some of the jobs. The non-renewable resource has an initial stock, which is replenished at a-priori known moments of time and in known quantities.

More formally, there is a single machine, a set of n jobs $\mathcal{J}$, and a non-renewable resource consumed by the jobs. The machine can process only one job at a time, and preemption of processing is not allowed. Each job J_j, $j \in \mathcal{J}$, has a processing time $p_j \in \mathbb{Z}_{\geq 1}$, a weight $w_j \in \mathbb{Z}_{\geq 0}$, and a resource requirement $a_j \in \mathbb{Z}_{\geq 0}$. The resource is supplied in q different moments in time, $0 = u_1 < u_2 < \ldots < u_q$; the scalar $\tilde{b}_\ell \in \mathbb{Z}_{\geq 1}$, for $\ell = 1, \ldots, q$, represents the quantity supplied at u_ℓ. A *schedule* σ specifies a starting time $S_j \in \mathbb{Z}_{\geq 0}$ for each job $j \in \mathcal{J}$, and it is *feasible* if (i) the jobs do not overlap in time and if (ii) at any time point t the total supply from the resource is at least the total request of those jobs starting not later than t, i.e., $\sum(\tilde{b}_\ell \mid \ell \in \{1, \ldots, q\} : u_\ell \leq t) \geq \sum(a_j \mid j \in \mathcal{J} : S_j \leq t)$. The objective is to minimize the total weighted completion time $\sum_j w_j C_j$, where $C_j = S_j + p_j$, S_j being the starting time of job J_j, $j \in \mathcal{J}$, in some feasible schedule.

We have assumed that $p_j \geq 1$ for all $j \in \mathcal{J}$, which is a slight restriction, but this helps to keep the presentation simple. In contrast, job j having zero resource requirement, i.e. $a_j = 0$, is possible.

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

Assumption 1 implies that there must exist a feasible solution (if every job starts not before u_q, the last supply date) and at least one job must start not before u_q (since all the supplies are positive).

The difficulty of this problem stems from the fact that distinct jobs may have different processing times and resource requirements, and sometimes jobs have to be delayed until a sufficient amount of the resource is supplied in order to start them.

The above problem is a generalization of the single machine scheduling problem with the total weighted job completion time objective, $1||\sum w_j C_j$, the latter being a very well understood and widely studied problem of scheduling theory. In fact, an optimal schedule can here be obtained by Smith's ratio rule [15], i.e., by scheduling the jobs in non-increasing w_j/p_j order. The structure of the polyhedron of feasible job completion times is also known [16]. More on the use of $1||\sum w_j C_j$ in scheduling theory can be found in [1, 4]. As even a single non-renewable resource constraint renders the problem intractable, see below, gaining more insight is a challenging research direction. As a practical application, consider a production line which processes a set of jobs, and the jobs require some raw materials that arrive over time. Finding a good ordering of the jobs with different material requirements, or processing times such that the weighted completion time of the jobs is minimized can be modelled by the scheduling problem described above.

1.1 Previous work

Scheduling problems with resource consuming jobs were introduced by Carlier [5], and Carlier and Rinnooy Kan [6]. Further results can be found in e.g., [10], [2, 3], [8], [11, 12]. In particular, Grigoriev et al. [10] studied the complexity of several variants and developed some constant ratio approximation algorithms. Briskorn et al. [2], and [3] studied scheduling problems with an initial inventory only, where some of the jobs produce, and other jobs consume the resources, the objective being the minimization of the inventory levels. In [11] a PTAS for scheduling resource consuming jobs with a single non-renewable resource and a constant number of supply dates was developed. Also, an FPTAS was devised for the special case with $q = 2$ supply dates. In contrast, if two resources are given, and $q = 2$, no FPTAS exists [12]. To our best knowledge, the only paper dealing with machine scheduling with a non-renewable resource and a min-sum type criterion (the average completion time) is by Gafarov et al. [8], who proved that minimizing the average completion time of the jobs ($1|nr = 1|\sum_j C_j$) is NP-hard in the ordinary sense when the number of supply dates is not a constant.

1.2 Results of the paper

We will prove by a reduction from the single machine scheduling problem with release dates and the average completion time objective that our problem is NP-

hard in the ordinary sense even if there is only one machine, one non-renewable resource, and among the jobs there is only one which requires a positive amount from the single resource. Moreover, our scheduling problem is NP-hard in the strong sense if the number of supply dates is part of the input (not fixed), and there are as many jobs requesting one unit each from the resource as the number of supply dates. These complexity results sharpen that of Gafarov et al. [8].

Our second result is a fully polynomial time approximation scheme (FPTAS) if the number of supply dates is $q = 2$, and there is only one non-renewable resource. The algorithm borrows ideas from the FPTAS described in [7, 12] and from results for the makespan minimization problem on parallel machines [14]. Since the scheduling problem is NP-hard in the strong sense if q is part of the input, there is no FPTAS in this general case, unless P=NP.

2 The NP-hardness proofs

We reduce the single machine scheduling problem with release dates and the average completion time objective ($1|r_j|\sum_j C_j$) to our scheduling problem. In this problem, jobs have release dates specifying their earliest possible start times. We will consider the subset of instances in which exactly one job has a positive release date, and all other jobs have a release date equal to 0. Rinnooy Kan [13] has shown that minimizing the total completion time in this subclass is already NP-hard. We use his result to establish the following theorem:

Theorem 1. *Minimizing the average completion time on a single machine with resource consuming jobs and one resource with $q = 2$ supply periods is NP-hard ($1|nr = 1, q = 2|\sum_j C_j$).*

Proof. We provide a transformation from the subclass of $1|r_j|\sum_j C_j$ in which there is precisely one job with a positive release date, while all other jobs have release date equal to 0, to our scheduling problem. Take any instance I in this subclass, the corresponding instance I' of our scheduling problem has one machine, one non-renewable resource, and the same set of jobs as I. Suppose J_k is the job with a positive release date in I. Then in I', the second supply date is $u_2 = r_k$, when an amount of one unit is supplied from the single resource, the first supply date is $u_1 = 0$ with an initial stock level 0. In I', all jobs have release date 0, but job J_k has a requirement of 1 unit from the non-renewable resource. Clearly, since the initial supply from the non-renewable resource is 0, J_k cannot start earlier than $u_2 = r_k$ in I'. Hence, the instances I and I' are equivalent in the sense that all jobs except J_k can start at time 0, and job J_k can start at r_k or later. Consequently, I has a solution with objective function value not greater than a given constant D if and only if I' has a solution with objective function value at most D. □

Rinnooy Kan also claims that the 3-PARTITION problem can be reduced to $1|r_j|\sum_j C_j$ by adapting the strong NP-hardness proof of Garey et al. [9] of the two-machine flow-shop scheduling problem with the average completion time objective ($F2||\sum_j C_j$). Recall that an instance of 3-PARTITION consists of $3q$

items each having a non-negative integer size, and a bound B such that each item size is between $B/4$ and $B/2$, and the question is whether the items can be grouped into q 3-element groups of total item size B each. By inspecting the latter proof, one can observe that in the claimed strong NP-hardness proof of $1|r_j|\sum_j C_j$ there are q jobs with q distinct release dates, and all other jobs have release dates 0. We can easily adapt the proof of Theorem 1 to this case. By this, we obtain the following result.

Theorem 2. *Minimizing the average completion time on a single machine with resource consuming jobs and one resource is NP-hard in the strong sense* ($1|nr = 1|\sum_j C_j$).

3 An FPTAS for $1|nr = 1, q = 2|\sum_j w_j C_j$

In this section we describe a fully polynomial time approximation scheme for the special case in which there is a single resource having an initial stock, and one additional supply at date $u_2 > 0$. Recall that an FPTAS for an optimization problem is a class of algorithms, which, for each $0 < \varepsilon < 1$, has an algorithm A_ε which runs in polynomial time in the size of the input and in $1/\varepsilon$, and produces a feasible solution of cost at most $(1+\varepsilon)$ times the cost of an optimal solution in case of minimization problems, and at least $(1-\varepsilon)$ times the cost of an optimal solution in case of maximization problems [14]. Our algorithm has a very simple structure and uses standard techniques, see [14]. The only interesting part is the transformation of the scheduling problem $1|nr = 1, q = 2|\sum_j w_j C_j$ to finding a shortest path in an appropriately defined graph, but the twist is that the shortest path computation is needed for finding a resource feasible solution, instead of computing the cost of the solution.

In order to get more insight into our scheduling problem, consider any instance in the subclass studied in this section, and any feasible solution to that instance. Since there are only two supply dates, u_1 and u_2, with $u_1 = 0$ with corresponding supply $\tilde{b}_1$, and $u_2 > 0$ with corresponding supply $\tilde{b}_2$, we know that the set of jobs is partitioned into $\mathcal{J}^1$ and $\mathcal{J}^2$, where the jobs in $\mathcal{J}^1$ use only the initial supply available at u_1, while the remaining jobs use that arriving at u_2 and the supply from $\tilde{b}_1$ left by the jobs in $\mathcal{J}^1$. Moreover, the jobs in $\mathcal{J}^2$ are scheduled after all the jobs in $\mathcal{J}^1$. Now we prove that there is an optimal solution with a special structure, and this will serve as a basis for our FPTAS.

Proposition 1. *The problem* $1|nr = 1, q = 2|\sum_j w_j C_j$ *always admits an optimal solution of the following structure:*

i) The set of jobs is partitioned into subsets $\mathcal{J}^1$ *and* $\mathcal{J}^2$*, the jobs in* $\mathcal{J}^1$ *use only the initial supply and are scheduled consecutively from time* u_1 *on, while the jobs in* $\mathcal{J}^2$ *use the supply left from* $\tilde{b}_1$ *and also the supply* $\tilde{b}_2$*, and are scheduled consecutively from the time point* $\max\{u_2, \sum_{j\in\mathcal{J}^1} p_j\}$.

ii) The jobs in both of $\mathcal{J}^1$ and $\mathcal{J}^2$ are scheduled in non-increasing w_j/p_j order, and there is at most one job in $\mathcal{J}^1$ which finishes after u_2, and if such a job exists, it starts before u_2.

Proof. Take any optimal solution, and let $\mathcal{J}^1$ consist of those jobs starting before u_2, and $\mathcal{J}^2$ contain the remaining jobs. Then, part i) holds, as one may easily verify. Moreover, there is at most one job in $\mathcal{J}^1$ which finishes after u_2, and this job must start before u_2 by definition. Further on, the jobs in $\mathcal{J}^2$ must be scheduled in non-increasing w_j/p_j order, otherwise the schedule is not optimal as a simple job-exchange argument shows (notice that after u_2, the resource is completely supplied, and there is nothing which could prevent such an ordering of the jobs in $\mathcal{J}^2$). Finally, suppose that the jobs in $\mathcal{J}^1$ are not scheduled in non-increasing w_j/p_j order, then again, a simple exchange argument shows that the schedule is not optimal, a contradiction, which proves part ii). □

In light of Proposition 1, we can express the objective function value of a schedule with a given partitioning $\mathcal{J}^1$ and $\mathcal{J}^2$ of the jobs (assuming that the jobs are indexed such that $w_1/p_1 \geq w_2/p_2 \geq \cdots \geq w_n/p_n$), as follows:

$$\sum_{j\in\mathcal{J}^1} w_j \left(\sum_{k\in\mathcal{J}^1, k\leq j} p_k \right) + \sum_{j\in\mathcal{J}^2} w_j \left(\max\left\{ u_2, \sum_{k\in\mathcal{J}^1} p_k \right\} + \sum_{k\in\mathcal{J}^2, k\leq j} p_k \right). \quad (1)$$

Notice that the first half of this expression calculates the weighted completion time of the jobs in $\mathcal{J}^1$ when scheduled in non-increasing w_j/p_j order, and the second half does the same for the jobs in $\mathcal{J}^2$.

In the following, denote, for any subset H of jobs, $p(H) := \sum_{j\in H} p_j$, and $w(H) := \sum_{j\in H} w_j$.

In the approximation algorithm for some $0 < \varepsilon < 1$, we will *round* some partial sums of the problem data, and *guess* the value of $\max\{u_2, p(\mathcal{J}^1)\}$. We define a *rounding function* $f(s)$ which assigns to any nonnegative number $s \geq 0$ a number from a discrete set as follows. Let $\Delta := 1 + \varepsilon/4n$. Then, we define

$$f(s) := \begin{cases} 0, & s = 0 \\ \Delta^{\lceil \log_\Delta s \rceil}, & s > 0 \end{cases},$$

where $\lceil x \rceil$ is the smallest integer $z \geq x$ for any $x \in \mathbb{R}$. Notice that $s \leq f(s) \leq s \cdot \Delta$ holds for any $s \geq 0$, since $\Delta > 1$. We will round iteratively sums of numbers, i.e., the iterative rounding of a sum of the form $x_1 + x_2 + \cdots + x_k$ is defined with the formula $g_t := f(x_t + g_{t-1})$ for $t = 1, \ldots, k$, where $g_0 := 0$.

Proposition 2. *If $x_t \geq 0$ for all $t = 1, \ldots, k$, then*

i) $x_1 + \cdots + x_t \leq g_t$, for $t = 1, \ldots, k$.

ii) $g_t \leq \Delta^t (x_1 + \cdots + x_t)$, for $t = 1, \ldots, k$.

Proof. For showing i), it is enough to note that the rounding function $f(\cdot)$ rounds up any value $s \geq 1$ to the least value $\Delta^k \geq s$, with $k \in \mathbb{Z}_{\geq 0}$.

The set of inequalities ii) is verified by induction on t. For $t = 1$, we have $g_1 = f(x_1) \leq \Delta x_1$, by using the definition of $f(\cdot)$. So assume that the inequality is proved for $t-1$, and we verify it for t: $g_t = f(x_t + g_{t-1}) \leq (x_t + g_{t-1})\Delta \leq (x_t + \Delta^{t-1}(\sum_{i=1}^{t-1} x_i))\Delta \leq \Delta^t(x_1 + \cdots + x_t)$, where the first inequality follows from the definition of $f(\cdot)$, the second from the induction hypothesis, and the third from $\Delta > 1$. □

This rounding scheme has been used e.g., in [14] (Section 0.5.1: Makespan on two identical machines).

We will guess the value of $\max\{u_2, p(\mathcal{J}^1)\}$ by picking a member from the set $G := \{u_2\Delta^z \ : \ z = 0, \ldots, \lceil \log_\Delta((u_2 + p(\mathcal{J}))/u_2)\rceil\}$. Notice that the largest value in G is bounded by $(u_2 + p(\mathcal{J}))\Delta$.

With this data, we build a directed graph D_δ for each $\delta \in G$. Firstly, we re-index the jobs such that $w_1/p_1 \geq w_2/p_2 \geq \cdots \geq w_n/p_n$. The nodes of D_δ represent all the distinct partitioning of the first t jobs, for all $t \in \{1, \ldots, n\}$, noting that the same node for a given t may represent several distinct partitionings of the first t jobs. We associate a vector with each node, that is, a node with the first t jobs has a vector $(t, P_1', PW_1', P_2', PW_2')$, where P_1' is the iterative rounding of the total processing times of those jobs assigned to the time period $[u_1, \delta]$ out of the first t jobs, and PW_1' is the iterative rounding of the total weighted completion times of these jobs. Likewise, P_2', and PW_2' are the iteratively rounded total processing times, and the iteratively rounded total weighted completion times, respectively, of those jobs assigned to the time period $[\delta, \infty)$, i.e., these jobs start at time δ. The unique source node (with zero in-degree) represents the empty schedule, when no job is chosen ($t = 0$). Then we gradually assign the jobs to time periods, and add more nodes to the graph, until all the jobs are assigned. Consider a node with vector $(t, P_1', PW_1', P_2', PW_2')$. It has two successor nodes, giving rise to two directed arcs. One of the successors corresponds to assigning job J_{t+1} to the time period $[u_1, \delta]$, and the other successor to assigning J_{t+1} to the time period $[\delta, \infty)$. The vectors associated with these two nodes can be easily computed from $(t, P_1', PW_1', P_2', PW_2')$: if J_{t+1} is assigned to the time period $[u_1, \delta]$, then the associated vector is $(t+1, f(P_1' + p_{t+1}), f(PW_1' + w_{t+1}(P_1' + p_{t+1})), P_2', PW_2')$, and the vector of the other node can be computed as $(t+1, P_1', PW_1', f(P_2' + p_{t+1}), f(PW_2' + w_{t+1}(\delta + P_2' + p_{t+1})))$. Clearly, the same node can be the successor of several other nodes. The arcs representing the assignment of job J_{t+1} to the time period $[u_1, \delta]$ have weight a_{t+1}, while all other arcs have weight 0, for all $t = 0, \ldots, n-1$. The nodes that contain a partitioning of all the n jobs are called *terminal nodes*. A terminal node with vector $(n, P_1', PW_1', P_2', PW_2')$ is *feasible* if it satisfies both of the following conditions: (i) $P_1' \leq \delta$, and (ii) the shortest path from the source node to this node has total arc length at most $\tilde{b}_1$. We say that D_δ *gives rise to a feasible solution* if and only if D_δ admits a feasible terminal node. The *value* of a feasible terminal node with vector $(n, P_1', PW_1', P_2', PW_2')$ is $PW_1' + PW_2'$. After these preliminaries, the complete FPTAS is as follows.

1. Determine the set G.

2. For each $\delta \in G$, compute the graph D_δ, and determine if D_δ gives rise to a feasible solution.

3. Choose D_{δ^*} which gives rise to a feasible solution with the smallest value overall.

4. Recover a schedule by following a shortest path in D_{δ^*} from the unique source node to a feasible terminal node of smallest rounded objective function value, and forming the sets $\mathcal{J}_*^1$ and $\mathcal{J}_*^2$ such that a job J_j is put in $\mathcal{J}_*^1$ if and only if J_j is assigned to time period $[u_1, \delta^*]$, and put into $\mathcal{J}_*^2$ otherwise.

Theorem 3. *The algorithm is an FPTAS for* $1|nr = 1, q = 2|\sum_j w_j C_j$.

Proof. Take any instance I of $1|nr = 1, q = 2|\sum_j w_j C_j$, and consider an optimal solution with the structure specified in Proposition 1. In particular, let $\mathcal{J}^1$ be the set of jobs starting before u_2. Let δ be the largest member of G with $\delta < \Delta^{n+1} \max\{u_2, p(\mathcal{J}^1)\}$ (since $u_2 \in G$, such a member exists). Since $\Delta^n \leq 1 + \varepsilon/2$ (using the fact that $(1 + x/n)^n \leq 1 + 2x$ for $0 \leq x \leq 1$, see [14]), we derive $\delta < \Delta^{n+1} \max\{u_2, p(\mathcal{J}^1)\} \leq (1 + \varepsilon) \max\{u_2, p(\mathcal{J}^1)\}$. Notice that since $\delta = u_2 \Delta^k$ for some $k \in \mathbb{Z}_{\geq 0}$, $\delta \geq \Delta^n p(\mathcal{J}^1)$ follows.

In D_δ consider a terminal node with associated vector $(n, P_1', PW_1', P_2', PW_2')$ which corresponds to the partitioning of jobs $\mathcal{J}^1, \mathcal{J}^2$ in the optimal solution. Then this must be a feasible terminal node of D_δ, since we started out from a feasible schedule, and $\delta \geq \Delta^n p(\mathcal{J}^1) \geq P_1' \geq p(\mathcal{J}^1)$, where the first inequality is from the preceding paragraph, while the second and third follow from Proposition 2. The value of this node is $PW_1' + PW_2'$. How much bigger is $PW_1' + PW_2'$ than the objective function value (1)? We compute

$$PW_1' + PW_2' < \Delta^n \left(\sum_{j \in \mathcal{J}^1} w_j \left(\sum_{k \in \mathcal{J}^1, k \leq j} p_k \right) + \sum_{j \in \mathcal{J}^2} w_j \left(\delta + \sum_{k \in \mathcal{J}^2, k \leq j} p_k \right) \right) <$$

$$\Delta^n \left(\sum_{j \in \mathcal{J}^1} w_j \left(\sum_{k \in \mathcal{J}^1, k \leq j} p_k \right) + \sum_{j \in \mathcal{J}^2} w_j \left((1 + \varepsilon) \max\{u_2, p(\mathcal{J}^1)\} + \sum_{k \in \mathcal{J}^2, k \leq j} p_k \right) \right)$$

$$\leq \Delta^n (1 + \varepsilon) \left(\sum_{j \in \mathcal{J}^1} w_j \left(\sum_{k \in \mathcal{J}^1, k \leq j} p_k \right) + \sum_{j \in \mathcal{J}^2} w_j \left(\max\left\{u_2, p(\mathcal{J}^1)\right\} + \sum_{k \in \mathcal{J}^2, k \leq j} p_k \right) \right)$$

$$\leq (1 + 2\varepsilon) \left(\sum_{j \in \mathcal{J}^1} w_j \left(\sum_{k \in \mathcal{J}^1, k \leq j} p_k \right) + \sum_{j \in \mathcal{J}^2} w_j \left(\max\left\{u_2, p(\mathcal{J}^1)\right\} + \sum_{k \in \mathcal{J}^2, k \leq j} p_k \right) \right),$$

where the first inequality follows from the properties of iterative rounding (Proposition 2), the second from the inequality $\delta < (1 + \varepsilon) \max\{u_2, p(\mathcal{J}^1)\}$ shown in the first paragraph of this proof, and the rest from the inequalities $\Delta^n \leq 1 + \varepsilon/2$ and $(1 + \varepsilon)(1 + \varepsilon/2) \leq 1 + 2\varepsilon$ for $0 \leq \varepsilon \leq 1$. Since the algorithm chooses a

feasible terminal node with smallest rounded value, we know that the value of the best solution found is also at most $(1+2\varepsilon)$ times the optimum.

The time complexity of the algorithm is determined by the cardinality of G, and the size of the graphs D_δ. The former set has $\lceil \log_\Delta((p(\mathcal{J})+u_2)/u_2)\rceil = \lceil \ln((p(\mathcal{J})+u_2)/u_2)/\ln\Delta\rceil$ elements, which is bounded by $CG := \lceil(\ln((p(\mathcal{J})+u_2)/u_2))(1+4n)/\varepsilon\rceil$, a polynomial in the size of the input and in $1/\varepsilon$, since $\ln(1+\varepsilon/4n) \geq \varepsilon/(\varepsilon+4n) \geq \varepsilon/(1+4n)$, and $\ln z \geq (z-1)/z$, see [14]. On the other hand, the number of nodes of D_δ is bounded by the number of distinct $(t, P'_1, PW'_1, P'_2, PW'_2)$ vectors. Since both of P'_1 and P'_2 take values from the set $\{0\} \cup \{\Delta^z \ : \ z \in \{0,\ldots,n+\lceil\log_\Delta p(\mathcal{J})\rceil\}\}$, and both of PW'_1 and PW'_2 take values from the set $\{0\} \cup \{\Delta^z \ : \ z \in \{0,\ldots,n+\lceil\log_\Delta w(\mathcal{J})(u_2+p(\mathcal{J}))\rceil\}\}$, we deduce that both of P'_1 and P'_2 may take at most $CP := n+\lceil\ln p(\mathcal{J})\rceil(1+4n)/\varepsilon$ distinct values, and that both of PW'_1 and PW'_2 may take at most $CPW := n+\lceil\ln w(\mathcal{J})(u_2+p(\mathcal{J}))\rceil(1+4n)/\varepsilon$ distinct values. Therefore, the number of nodes is bounded by a polynomial in the size of the input, and in $1/\varepsilon$. Since the out-degree of each non-terminal node is 2, we know that the size of D_δ is also polynomial in the size of the input and in $1/\varepsilon$, and thus the shortest path to all the terminal nodes can be computed in polynomial time in the size of the input (in linear time in the size of the graph, since D_δ is acyclic), and in $1/\varepsilon$. Therefore, we have an FPTAS with running time $O(CG \cdot n \cdot CP^2 \cdot CPW^2)$. □

4 Final remarks

It is tempting to extend the FPTAS for a fixed number of supply dates $q \geq 2$. However, a major obstacle is how to check the resource-feasibility of the solution.

Acknowledgments

The author is grateful for a referee for careful reading and several comments that helped to improve the paper. The research of Tamás Kis has been supported by the János Bolyai research grant BO/00412/12/3 of the Hungarian Academy of Sciences, and by the OTKA grant K112881.

References

[1] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, J. Weglarz, Handbook on Scheduling, From Theory to Applications, Springer-Verlag Berlin Heidelberg, 2007. ISBN: 978-3-540-28046-0.

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

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

[4] P. Brucker, Scheduling Algorithms, Springer-Verlag Berlin Heidelberg, 2007, ISBN: 978-3-540-69515-8.

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

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

[7] M. Drótos, T. Kis, Scheduling of inventory releasing jobs to minimize a regular objective function of delivery times, Journal of Scheduling, 16 (2013) 337–346.

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

[9] M.R. Garey, D.S. Johnson, Ravi Sethi, The complexity of flowshop and jobshop scheduling, Mathematics of Operations Research, 2 (1976) 117–129.

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

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

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

[13] A.H.G. Rinnooy Kan, Machine scheduling problems : classification, complexity and computations, Martinus Nijhoff, The Hague, 1976.

[14] P. Schuurman, G. Woeginger, Approximation schemes-a tutorial, In: R. Moehring, C. Potts, A. Schulz, G. Woeginger, L.A. Wolsey. (eds.) Lectures on Scheduling. Forthcoming. (url: www.win.tue.nl/ gwoegi/papers/ptas.pdf)

[15] W.E. Smith, Various optimizers for single-stage production, Naval Res. Logist. Quart., 3 (1956) 59–66.

[16] M. Queyranne, Structure of a simple scheduling polyhedron, Mathematical Programming, 58 (1993) 263–285.