Contents lists available at ScienceDirect

Mathematical Social Sciences

Single machine scheduling problems with financial resource constraints: Some complexity results and properties

Evgeny R. Gafarov [a], Alexander A. Lazarev [a,b], Frank Werner [c,*]

[a] *Institute of Control Sciences of the Russian Academy of Sciences, Profsoyuznaya str. 65, Moscow, 117997, Russia*
[b] *Higher School of Economics – State University, Myasnitskaya str. 20, Moscow, 101990, Russia*
[c] *Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, PSF 4120, 39016 Magdeburg, Germany*

ARTICLE INFO

ABSTRACT

We consider single machine scheduling problems with a non-renewable resource. These types of problems have not been intensively investigated in the literature so far. For several problems of these types with standard objective functions (namely the minimization of makespan, total tardiness, number of tardy jobs, total completion time and maximum lateness), we present some complexity results. Particular attention is given to the problem of minimizing total tardiness. In addition, for the so-called budget scheduling problem with minimizing the makespan, we present some properties of feasible schedules.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

In a resource-constrained scheduling problem, one wishes to schedule the jobs in such a way that the given resource constraints are fulfilled and a given objective function attains its optimal value. We consider single machine scheduling problems with a non-renewable resource. For example, money or fuel provides natural examples of such a non-renewable resource. Such problems with a non-renewable resource are also referred to as *financial scheduling* problems. They are e.g. important for government-financed organizations. To illustrate, it is usual for government-financed organizations that they cannot or that they do not want to begin a project before receiving all necessary payments. For example, consider a road-building company. Each of the projects of such a company is the construction of one road. The company can work only on one project at the same time due to resource constraints. However, at the beginning of a project, it must buy materials, it has to pay subcontractors, etc. Usually, the company wants to be *sure* that it will receive all money from the government in time.

The research in the area of scheduling problems with a non-renewable resource is rather limited. In Carlier and Rinnooy Kan (1982), some polynomially bounded algorithms are presented for scheduling problems with precedence constraints (not restricted to single machine problems). Some results for preemptive scheduling of independent jobs on unrelated parallel machines have been presented in Slowinski (1984). Toker et al. (1991) have shown that the problem of minimizing the makespan with a unit supply of a resource at each time period is polynomially solvable. Janiak (1986); Janiak et al. (2000) have considered single machine problems, in which the processing times or the release times depend on the consumption of a non-renewable resource.

The problems under consideration can be formulated as follows. We are given a set $N = \{1, 2, \ldots, n\}$ of n independent jobs that must be processed on a single machine. Preemptions of a job are not allowed. The machine can handle only one job at a time. All the jobs are assumed to be available for processing at time 0. For each job j, $j \in N$, a processing time $p_j \geq 0$ and a due date d_j are given. In addition, we have one non-renewable resource G (e.g. money, fuel, etc.) and a set of times $\{t_0, t_1, \ldots, t_y\}$, $t_0 = 0$, $t_0 < t_1 < \cdots < t_y$, of earnings of the resource. At each time t_i, $i = 0, 1, \ldots, y$, we receive an amount $G(t_i) \geq 0$ of the resource. For each job $j \in N$, a consumption $g_j \geq 0$ of the resource arises when the job is started. Thus, we have

$$\sum_{j=1}^{n} g_j = \sum_{i=0}^{y} G(t_i).$$

Let S_j be the starting time of the processing of job j. A schedule $S = (S_{j_1}, S_{j_2}, \ldots, S_{j_n})$ describes the order of processing the jobs: $\pi = (j_1, j_2, \ldots, j_n)$. Such an order is uniquely determined by a permutation (sequence) of the jobs of set N. A schedule $S = (S_{j_1}, S_{j_2}, \ldots, S_{j_n})$ is feasible, if the machine processes no more than one job at a time and the resource constraints are fulfilled, i.e., for each $i = 1, 2, \ldots, n$, we have

$$\sum_{k=1}^{i} g_{j_k} \leq \sum_{\forall l: t_l \leq S_{j_i}} G(t_l).$$

* Corresponding author. Tel.: +49 391 6712025; fax: +49 391 6711171.
E-mail addresses: axel73@mail.ru (E.R. Gafarov), jobmath@mail.ru (A.A. Lazarev), frank.werner@ovgu.de, frank.werner@mathematik.uni-magdeburg.de (F. Werner).

Moreover, we will call the sequence π a schedule as well since one can compute $S = (S_{j_1}, S_{j_2}, \ldots, S_{j_n})$ in $O(n)$ time by applying a list scheduling algorithm to the sequence π. Then $C_{j_k}(\pi) = S_{j_k} + p_{j_k}$ denotes the completion time of job j_k in schedule π. If $C_j(\pi) > d_j$, then job j is tardy and we have $U_j = 1$, otherwise $U_j = 0$. Moreover, let $T_j(\pi) = \max\{0, C_j(\pi) - d_j\}$ be the tardiness of job j in schedule π. We denote by $C_{\max} = C_{j_n}(\pi)$ the makespan of schedule π and by $L_j(\pi) = C_j(\pi) - d_j$ the lateness of job j in π. In this paper, notation $\{\pi\}$ denotes the set of jobs contained in sequence π. Notation $i \in \pi$ means $i \in \{\pi\}$.

The single machine problem with a non-renewable resource and the minimization of the makespan $C_{\max}$ is denoted as $1|NR|C_{\max}$. The corresponding problem of minimizing total tardiness $\sum_{j=1}^{n} T_j(\pi)$ is denoted as $1|NR|\sum T_j$. In a similar way, we use the notations $1|NR|\sum C_j$, $1|NR|\sum U_j$ and $1|NR|L_{\max}$ for the problems of minimizing total completion time, the number of tardy jobs and maximum lateness.

The rest of this paper is organized as follows. In Section 2, we present some complexity results for the single machine problems mentioned above. In Section 3, we consider problem $1|NR|\sum T_j$ and prove that this problem is *NP*-hard even for the special case of equal processing times. We also investigate some properties of further special cases of this problem. The budget scheduling problem of minimizing the makespan is considered in Section 4.

2. Some complexity results

In Table 1, we summarize the complexity results derived in this section. In this table, we refer to the partition and 3-partition problems which are as follows.

Partition problem.

A set $N = \{b_1, b_2, \ldots, b_n\}$ of numbers is given with $b_1 \geq b_2 \geq \cdots \geq b_n > 0$ and $b_i \in Z_+$, $i = 1, 2, \ldots, n$. Does there exist a subset $N' \subseteq N$ such that

$$\sum_{i \in N'} b_i = \frac{1}{2} \sum_{i=1}^{n} b_i?$$

It is known that this problem is *NP*-complete in the ordinary sense (Garey and Johnson, 1979). There exists a pseudo-polynomial algorithm for this problem (Garey and Johnson, 1979).

The 3-partition problem is to decide whether a given set of integers can be partitioned into triplets that all have the same sum. More precisely:

3-*Partition problem.*

A set $N = \{b_1, b_2, \ldots, b_n\}$ of $n = 3m$ positive integers is given, where $\sum_{i=1}^{n} b_j = mB$ and $\frac{B}{4} < b_j < \frac{B}{2}$, $j = 1, 2, \ldots, n$. Does there exist a partition of N into m subsets $N_1, N_2, \ldots, N_m$ such that each subset contains exactly three numbers and the sum of the numbers in each subset is equal, i.e.,

$$\sum_{b_j \in N_1} b_j = \sum_{b_j \in N_2} b_j = \cdots = \sum_{b_j \in N_m} b_j = B?$$

The 3-partition problem is *NP*-complete in the strong sense (Garey and Johnson, 1979).

Theorem 1. *The problems* $1|NR|C_{\max}$, $1|NR, d_j = d|\sum T_j$, $1|NR|\sum U_j$ *and* $1|NR|L_{\max}$ *are NP-hard in the strong sense, and the problem* $1|NR|\sum C_j$ *is NP-hard.*

Proof. For all problems except the last one, we give reductions from the 3-partition problem.

1. First, we consider the problem $1|NR|C_{\max}$. Given the instance of the 3-partition problem, we define an instance of problem

$1|NR|C_{\max}$ as follows. There are n jobs with $g_j = p_j = b_j$, $j = 1, 2, \ldots, n$. The times of earnings of the resource are given by $\{t_0, t_1, \ldots, t_{m-1}\} = \{0, B, 2B, \ldots, (m-1)B\}$ and $G(t_0) = G(t_1) = \cdots = G(t_{m-1}) = B$. It is obvious that $C_{\max} = mB$ if and only if the answer for the 3-partition problem is "YES". In this case, there are no idle times in an optimal schedule.

2. For the problems $1|NR, d_j = d|\sum T_j$, $1|NR|\sum U_j$ and $1|NR|L_{\max}$, we define in addition to the above data $d_j = d = mB$ for $j = 1, 2, \ldots, n$. Then $\sum T_j = 0$ holds if and only if the answer for the 3-partition problem is "YES". Similarly, we have $\sum U_j = 0$ and $L_{\max} = 0$ if and only if the answer for the 3-partition problem is "YES".

3. Finally, we prove the NP-hardness of the problem $1|NR|\sum C_j$. It is known that the problem $1|r_j|\sum C_j$ is *NP*-hard in the strong sense (Lenstra et al., 1977).

   The reduction from the problem $1|r_j|\sum C_j$ to the problem $1|NR|\sum C_j$ can be described as follows. For the problem $1|r_j|\sum C_j$, there are given n jobs with $r_1 \leq r_2 \leq \cdots \leq r_n$. We consider the following instance of the problem $1|NR|\sum C_j$. There are n jobs and the times of earnings of the resource are given by $t_1 = r_1, t_2 = r_2, \ldots, t_n = r_n$. Moreover, let $G(t_i) = g_i = 10^{i-1}$, $i = 1, 2, \ldots, n$. If there are several jobs $l, l+1, \ldots, m-1, m$, with the same release date, then we assume $G(t_l) = \sum_{j=l}^{m} g_j = \sum_{j=l}^{m} 10^{j-1}$. Note that we have

   $$\sum_{i=1}^{k} g_k = 1 \cdot \frac{10^k - 1}{10 - 1} < 10^k = g_{k+1}, \quad 1 \leq k < n,$$

   i.e., for any feasible schedule, we have $S_{k+1} \geq t_{k+1} = r_{k+1}$.
   It is obvious that there is a one-to-one correspondence between feasible schedules for both problems and the corresponding objective function values. Therefore, the problem $1|NR|\sum C_j$ is *NP*-hard. Here we proved only NP-hardness in the ordinary sense, since the maximal length MAX(I) of the input of this special instance I is not bounded from above by a polynomial in the length of the input of the original instance. □

Theorem 2. *The problem* $1|NR, d_j = d|\sum T_j$ *is not in APX, where APX is the class of optimization problems that allow polynomial-time approximation algorithms with an approximation ratio bounded by a constant.*

Proof. For the proof, it suffices to note that the special case of the problem $1|NR, d_j = d|\sum T_j$ with the optimal value $\sum T_j = 0$ is *NP*-hard in the strong sense. □

3. Problem $1|NR|\sum T_j$

The classical scheduling problem $1 \| \sum T_j$ is *NP*-hard in the ordinary sense (Du and Leung, 1990; Lazarev and Gafarov, 2006). A dynamic programming algorithm of pseudo-polynomial time complexity $O(n^4 \sum p_j)$ has been proposed by Lawler (1977). The state-of-the-art algorithms by Szwarc et al. (1999) can solve special instances (Potts and Van Wassenhove, 1982) of this problem for $n \leq 500$ jobs. Some polynomially solvable special cases for the problem $1 \| \sum T_j$ have been given, e.g., by Lazarev and Werner (2009).

We start this section with the consideration of sub-problem $1|NR, p_j = p|\sum T_j$ with equal processing times. Then in the second subsection, we give a proof of *NP*-hardness for the special case $1|NR, d_j = d, g_j = g|\sum T_j$. In the last subsection, we present some properties of problem $1|NR|\sum T_j$ with arbitrary processing times.

3.1. Special case $1|NR, p_j = p|\sum T_j$

For this special case, we can present the following trivial result.

Table 1
An overview on the complexity results.

Obj. func.	Special case	*NP*-hardness, solution algorithm
$C_{\max}$		*NP*-hard in the strong sense (reduction from the 3-partition problem, see Theorem 1)
$\sum U_j$		*NP*-hard in the strong sense (Theorem 1)
$L_{\max}$		*NP*-hard in the strong sense (Theorem 1)
$\sum C_j$		*NP*-hard (reduction from problem $1|r_j|\sum C_j$, see Theorem 1)
$\sum T_j$	$d_j = d$	*NP*-hard in the strong sense (reduction from the 3-partition problem, see Theorem 1)
$\sum T_j$	$g_j = g$	*NP*-hard (problem $1 \parallel \sum T_j$ is a sub-case)
$\sum T_j$	$p_j = p$	*NP*-hard (reduction from the partition problem, see Theorem 4) (in addition: $1 \parallel \sum T_j \propto 1|p_j = p, G(t) = 1, t = 0, 1, \ldots | \sum T_j$)[a]
$\sum T_j$	$d_j = d, \ g_j = g$	*NP*-hard (reduction from the partition problem, see Theorem 6)
$\sum T_j$	$p_j = p, g_1 \le g_2 \le \cdots \le g_n, d_1 \le d_2 \le \cdots \le d_n$	Polynomially solvable (see Lemma 1), optimal solution: $\pi^* = (1, 2, \ldots, n)$

[a] The notation $P1 \propto P2$ means that problem $P1$ can be polynomially reduced to $P2$.

Remark 1. For the problem $1|NR, p_j = p| \sum T_j$, there exists an optimal schedule which has the structure $\pi = (\pi_1, \pi_2, \ldots, \pi_y)$, where the jobs in the partial schedule π_i, $i = 1, 2, \ldots, y$, (for the definition of y, see the description of this problem in Section 1) are processed in EDD (earliest due date) order.

In addition, we obtain the following polynomially solvable case.

Lemma 1. *For the special case of the problem $1|NR, p_j = p| \sum T_j$ with $g_1 \le g_2 \le \cdots \le g_n$, $d_1 \le d_2 \le \cdots \le d_n$, the schedule $\pi^* = (1, 2, \ldots, n)$ is optimal.*

Proof. The proof of polynomial solvability of this special case is trivial. For the two sequences $\pi = (\pi_1, k, l, \pi_2)$ and $\pi' = (\pi_1, l, k, \pi_2)$, where $l < k$, we have

$$\sum_{j=1}^{n} T_j(\pi) - \sum_{j=1}^{n} T_j(\pi') \ge (T_k(\pi) + T_l(\pi)) - (T_k(\pi') + T_l(\pi'))$$

$$\ge 0$$

since $C_l(\pi') \le C_k(\pi)$, $C_k(\pi') \le C_l(\pi)$ and $d_l \le d_k$. □

Next, we consider a more specific situation, namely a sub-problem denoted as $1|NR : \alpha_t = 1, p_j = p| \sum T_j$ (see below). After proving *NP*-hardness of this special case, we consider another special case denoted as $1|NR, G(t) = M, p_j = p| \sum T_j$ and derive a relation between these two sub-problems. Then we give a proof of *NP*-hardness for problem $1|NR, p_j = p| \sum T_j$.

Now we consider the situation, where the times of earnings of the resource are given by $\{t_1, t_2, \ldots, t_y\} = \{1, 2, \ldots, \sum g_j\}$, $t_1 = 1, t_2 = 2, \ldots, t_y = \sum g_j$, and $G(t_i) = 1$ for $i = 1, 2, \ldots, y$. This condition is denoted as $\alpha_t = 1$ (Toker et al., 1991). Therefore, we can denote this problem as $1|NR : \alpha_t = 1, p_j = p| \sum T_j$.

Theorem 3. *The problem $1|NR : \alpha_t = 1, p_j = p| \sum T_j$ is NP-hard.*

Proof. We give the following reduction from the problem $1 \parallel \sum T_j$. Given an instance of the problem $1 \parallel \sum T_j$ with processing times p'_j and due dates d'_j for $j = 1, 2, \ldots, n$, we construct an instance of problem $1|NR : \alpha_t = 1, p_j = p| \sum T_j$ as follows. Let $g_j = p'_j, p_j = 0$ and $d_j = d'_j$ for $j = 1, 2, \ldots, n$. Then both problems are equivalent. □

It can be noted that the special case $1|NR : \alpha_t = 1, p_j = 0| \sum T_j$ can be solved in $O(n^4 \sum g_j)$ time by Lawler's algorithm (Lawler, 1977) since we obtain a problem $1 \parallel \sum T_j$ with processing times g_j. Later in Theorem 4, we will present another *NP*-hardness proof for the problem $1|NR : p_j = p| \sum T_j$. The reason for this is as follows. Let I be an instance of the problem $1|NR : p_j = p| \sum T_j$ and x be a string of the form

"$p, d_1, d_2, \ldots, d_n, g_1, g_2, \ldots, g_n, t_0, t_1,$
   $\ldots, t_y, G(t_0), G(t_1), \ldots, G(t_y)$",

encoding the instance I under a reasonable encoding scheme e. According to the definition in Garey and Johnson (1979), we have LENGTH$[I] = n + y$. In fact, the string x consists of $2n + 2y + 3$ numbers. However, if we consider the problem $1|NR : \alpha_t = 1, p_j = 0| \sum T_j$ as a special case of the problem $1|NR : p_j = p| \sum T_j$ and use the same encoding scheme, then LENGTH$[I] = n + y = n + \sum g_j$, i.e., the length of the input is pseudo-polynomial. Since, as mentioned above, the problem $1|NR : \alpha_t = 1, p_j = 0| \sum T_j$ can be solved in $O(n \sum g_j)$ time by Lawler's algorithm, the complexity of this algorithm would polynomially depend on the input length LENGTH$[I] = n + \sum g_j$. For this reason, we consider sub-problem $1|NR : \alpha_t = 1, p_j = p| \sum T_j$ as a separate problem and use the encoding scheme e', in which we present an instance as a string

"$p, d_1, d_2, \ldots, d_n, g_1, g_2, \ldots, g_n$",

i.e., LENGTH$[I] = n$.

We consider the sub-case of problem $1|NR, p_j = p| \sum T_j$, where the times of earnings of the resource are given by $t_1 = M, t_2 = 2M, \ldots, t_n = nM$ and $G(t_i) = M$ for all $i = 1, 2, \ldots, n$, where $M = \frac{\sum g_j}{n}$ such that $M \in Z_+$. We denote this special case by $1|NR, G(t) = M, p_j = p| \sum T_j$.

Two instances of problems $1|NR : \alpha_t = 1, p_j = p| \sum T_j$ and $1|NR, G(t) = M, p_j = p| \sum T_j$ are called corresponding, if all parameters $d_j, p_j, g_j, j = 1, 2, \ldots, n$, for the two instances are the same.

Now we investigate a relation between the values of the objective function $\sum T_j$ for two corresponding instances. First, we show that these two instances can have different optimal sequences (see Lemma 2). Then we consider the special case of $d_j = 0$ for $j = 1, 2, \ldots, n$ such that the objective function $\sum T_j$ turns into the special case of $\sum C_j$, and we investigate the difference between the function values of the same sequence for the two problems mentioned above (see Lemmas 3 and 4).

Lemma 2. *There exist two corresponding instances of the problems $1|NR : \alpha_t = 1, p_j = p| \sum T_j$ and $1|NR, G(t) = M, p_j = p| \sum T_j$ which have different optimal schedules.*

Proof. We consider an instance with $n = 2$ jobs and $p_1 = p_2 = 1$, $g_1 = 1$, $g_2 = 5$, $d_1 = 7$, $d_2 = 6$. For problem $1|NR : \alpha_t = 1, p_j = p| \sum T_j$, we have $\sum T_j(\pi^1) = 0$ and $\sum T_j(\pi^2) = 1$, where $\pi^1 = (2, 1)$ and $\pi^2 = (1, 2)$. On the other hand, for the problem $1|NR, G(t) = M, p_j = p| \sum T_j$, we have $\sum T_j(\pi^1) = 2$ and $\sum T_j(\pi^2) = 1$. Thus, the above two instances have different optimal schedules. □

Now, let $d_j = 0$ for $j = 1, 2, \ldots, n$. For two corresponding instances of problems $1|NR : \alpha_t = 1, p_j = 1| \sum C_j$ and $1|NR, G(t) = M, p_j = 1| \sum C_j$, let $C_j(\pi)$ be the completion time of job j according to the job sequence π for the first problem and $C'_j(\pi)$ be the completion time of the same job according to π for the second problem. Then we can prove the following lemma.

Lemma 3. *For two corresponding instances of the problems* $1|NR : \alpha_t = 1, p_j = 1| \sum C_j$ *and* $1|NR, G(t) = M, p_j = 1| \sum C_j$, *we have*

$$\frac{\sum_{j=1}^{n} C'_j(\pi)}{\sum_{j=1}^{n} C_j(\pi)} < 2. \tag{1}$$

Proof. First, we show that inequality

$$C'_j(\pi) - C_j(\pi) \leq M - 1 \tag{2}$$

holds for each job $j, j = 1, 2, \ldots, n$. We denote $S'_j(\pi) = C'_j(\pi) - p_j = C'_j(\pi) - 1$.

Let $\pi = (1, 2, \ldots, i, i + 1, \ldots, j, \ldots, n)$. Then there exists a natural number k such that

$$S_{i-1}(\pi) < (k - 1)M \leq S_u(\pi) < kM$$

for $u = i, i + 1, \ldots, j$. In addition, it is obvious that

$$S'_u(\pi) < (k + 1)M$$

for $u = i, i + 1, \ldots, j$. If the jobs $i, i + 1, \ldots, j$ are processed from time kM in schedule π for problem $1|NR, G(t) = M, p_j = 1| \sum C_j$, then $S_i(\pi) > (k - 1)M$ and

$$S'_j(\pi) - kM = \sum_{l \in \{i, i+1, \ldots, j-1\}} p_l = j - i < S_j(\pi) - (k - 1)M.$$

Thus, inequality (2) holds.

Moreover, we have $C_{\max}(\pi) \geq nM + 1$ since $t = nM$ is the last time of earnings of the resource, i.e., nM is a lower bound for $C_{\max}$ and $C'_{\max}$. Hence, we get

$$\sum_{j=1}^{n} C'_j(\pi) - \sum_{j=1}^{n} C_j(\pi) \leq n(M - 1) < C_{\max}(\pi) < \sum_{j=1}^{n} C_j(\pi).$$

Therefore, inequality (1) holds. $\quad\square$

Lemma 4. *There exists an instance of the problems* $1|NR : \alpha_t = 1, p_j = 1| \sum C_j$ *and* $1|NR, G(t) = M, p_j = 1| \sum C_j$ *for which we have*

$$\frac{\sum_{j=1}^{n} C'_j(\pi)}{\sum_{j=1}^{n} C_j(\pi)} \approx 2 - \frac{1}{n}.$$

Proof. Let us consider the following instance. There are given n jobs with $g_j = 1, j = 1, 2, \ldots, n - 1$ and $g_n = nM - (n - 1)$, where $M = n^3 \sum_{j \in N \setminus \{n\}} g_j$. For schedule $\pi = (1, 2, \ldots, n)$, using inequality (2) in the proof of Lemma 3, we obtain

$$\sum_{j=1}^{n-1} C'_j(\pi) - \sum_{j=1}^{n-1} C_j(\pi) = (n - 1)(M - 1).$$

Moreover, we have

$$C_n = C_{\max} = C'_n = nM + 1.$$

This yields

$$\sum_{j=1}^{n} C_j(\pi) = 2 + 3 + \cdots + n + nM + 1$$

$$= \frac{n(n + 1)}{2} - 1 + nM + 1.$$

Let n be fixed. Then we obtain

$$\lim_{M \to \infty} \frac{\sum_{j=1}^{n} C'_j(\pi)}{\sum_{j=1}^{n} C_j(\pi)} = \lim_{M \to \infty} \frac{(n - 1)(M - 1) + \frac{n(n+1)}{2} + nM}{\frac{n(n+1)}{2} + nM}$$

$$= \frac{2n - 1}{n}$$

$$= 2 - \frac{1}{n}. \quad\square$$

Now we consider the sub-case of the problem $1|NR, p_j = p| \sum T_j$, where the number of times of earnings of the resource given by $t_0, t_1, \ldots, t_y$ is less than or equal to n, i.e., $y \leq n$ (we remind the discussion after the proof of Theorem 3).

Theorem 4. *The special case* $1|NR, p_j = p| \sum T_j$, *where the number of times of earnings of the resource given by* $t_0, t_1, \ldots, t_y$ *is less than or equal to* n, *is NP-hard.*

Proof. The reduction is done from the partition problem. We consider an instance of the scheduling problem with n jobs and $p_j = 0, g_j = b_j$ for all $j = 1, 2, \ldots, n$. Two times of earnings of the resource $t_0 = 0$ and $t_1 = M$ are given, where $M = (n \sum_{j=1}^{n} b_j)^2$ and $G(t_0) = G(t_1) = \frac{1}{2} \sum_{j=1}^{n} b_j$. In addition, the due dates $d_j = M - g_j$ for $j = 1, 2, \ldots, n$ are given.

Then all optimal schedules have the form $\pi = (E, F)$ with the following property. For all jobs $j \in E$, we have

$$C_j(\pi) = 0, \qquad T_j(\pi) = 0, \qquad \sum_{j \in E} g_j \leq \frac{1}{2} \sum_{j=1}^{n} b_j$$

and for all jobs $j \in F$, we have

$$C_j(\pi) = M, \qquad T_j(\pi) = M - d_j = g_j.$$

Then the instance of the partition problem has an answer "YES" if and only if in an optimal schedule π of the problem $1|NR, p_j = p| \sum T_j$, we have

$$\sum_{j=1}^{n} T_j(\pi) = \frac{1}{2} \sum_{j=1}^{n} b_j. \quad\square$$

As an immediate corollary of Theorem 4 we obtain that the general problem $1|NR, p_j = p| \sum T_j$ without the above additional condition is NP-hard which is the result presented in Table 1.

3.2. Special case $1|NR, d_j = d, g_j = g| \sum T_j$

In this subsection, a proof of NP-hardness for the special case $1|NR, d_j = d, g_j = g| \sum T_j$ is presented.

We give the following reduction from the partition problem. Denote $M = (n \sum_{j=1}^{n} b_j)^n$. Let us consider the following instance with the set of jobs $N = \{1, 2, \ldots, 2n + 1\}$:

$$\begin{cases} p_{2n+1} = 1, \\ p_{2i} = M^{n-i+1}, & i = 1, 2, \ldots, n, \\ p_{2i-1} = p_{2i} + b_i, & i = 1, 2, \ldots, n, \\ d = \displaystyle\sum_{i=1}^{n} p_{2i} + \frac{1}{2}\sum b_j, \\ g = 1, \\ t_0 = 0, & G(t_0) = n, \\ t_1 = d, & G(t_1) = 1, \\ t_2 = t_1 + \sum b_j + 1, & G(t_2) = 1, \\ t_3 = t_2 + p_{2n} + b_n, & G(t_3) = 1, \\ \cdots & \cdots \\ t_i = t_{i-1} + p_{2(n-i+3)} + b_{n-i+3}, & G(t_i) = 1, \\ \cdots & \cdots \\ t_{n+1} = t_n + p_4 + b_2, & G(t_{n+1}) = 1. \end{cases} \tag{3}$$

It is obvious that there are at least $n+1$ tardy jobs in any feasible schedule. We define a *canonical schedule* as a schedule of the form

$$(V_{1,1}, V_{2,1}, \ldots, V_{i,1}, \ldots, V_{n,1}, 2n+1,$$
$$V_{n,2}, \ldots, V_{i,2}, \ldots, V_{2,2}, V_{1,2}),$$

where

$$\{V_{i,1}, V_{i,2}\} = \{2i-1, 2i\}, \quad i = 1, 2, \ldots, n.$$

Moreover, let $\pi = (E, F)$ and for two partial schedules E and F, we have $|\{E\}| = n$ and $|\{F\}| = n+1$. Note that in any canonical schedule, all jobs in sub-sequence F are tardy, the last job in sub-sequence E can be tardy or on-time while all other jobs in sub-sequence E are on-time.

Theorem 5. *For instance* (3), *there exists an optimal schedule which is canonical.*

Proof. Without loss of generality, assume that $n > 2$ and $\sum b_j > 2$.

(1) Let us first prove that job $V_{1,2}$ (i.e., one of the jobs 1 or 2) is the last job in an optimal schedule. Assume that there exists an optimal schedule $\pi = (\pi_1, V_{1,1}, \pi_2, V_{1,2}, \pi_3, j)$. Denote $P_1 = \sum_{i \in N \setminus \{V_{1,1}, V_{1,2}\}} p_i$. It is obvious that $C_{V_{1,2}}(\pi) > 2M^n > t_{n+1}$. Moreover, $2M^n - t_{n+1} > 2P_1$.
Then, for schedule $\pi' = (\pi_1, V_{1,1}, \pi_2, \pi_3, j, V_{1,2})$, we have

$$\sum_{j=1}^{2n+1} T_j(\pi) - \sum_{j=1}^{2n+1} T_j(\pi') \geq (T_{V_{1,2}}(\pi) - T_{V_{1,2}}(\pi'))$$
$$+ (T_j(\pi) - T_j(\pi'))$$
$$> -P_1 + (C_{V_{1,2}}(\pi) - t_{n+1})$$
$$> -P_1 + 2M^n - t_{n+1} > P_1 > 0.$$

Thus, schedule π is not optimal, and the job $V_{1,2}$ is the last job in an optimal schedule.

(2) We prove that in an optimal schedule $\pi = (E, F)$, the job $V_{1,1}$ belongs to sub-sequence E with $|\{E\}| = n$. Assume that $\pi = (\pi_1, \pi_2, V_{1,1}, \pi_3, V_{1,2})$, where $\pi_1 = E$. It is obvious that $d - P_1 > 0$ and $C_{V_{1,2}}(\pi) - t_{n+1} > 2M^n + d - t_{n+1} > 2M^n - P_1$.
Let us consider schedule $\pi' = (V_{1,1}, \pi_1, \pi_2, \pi_3, V_{1,2})$. For each job $i \in \{\pi_1\} \bigcup \{\pi_2\} \bigcup \{\pi_3\}$, we have $T_i(\pi') - T_i(\pi) < P_1$. Moreover, $T_{V_{1,2}}(\pi) - T_{V_{1,2}}(\pi') > M^n - P_1$. Then

$$\sum_{j=1}^{2n+1} T_j(\pi) - \sum_{j=1}^{2n+1} T_j(\pi') > M^n - P_1 - (2n-1)P_1 > 0,$$

since $M^n > \frac{1}{2}M \cdot P_1 > 2nP_1$. Thus, in an optimal schedule $\pi = (E, F)$, the job $V_{1,1}$ belongs to sub-sequence E with $|\{E\}| = n$.

(3) Let us show that there exists an optimal schedule $\pi = (E, F)$, where the job $V_{1,1}$ is the first job.
  (3.1) First, we prove that in an optimal schedule, only one of the jobs $V_{2,1}$ and $V_{2,2}$ can be in sub-sequence E. In contradiction, without loss of generality, assume that $\pi = (V_{2,1}, V_{2,2}, \pi_1, V_{1,1}, j, \pi_2, V_{1,2})$, where $\{V_{2,1}, V_{2,2}\} \bigcup \{\pi_1\} \bigcup \{V_{1,1}\} = \{E\}$ and j is the first job in sub-sequence F. From the previous item, it is obvious that there exists an optimal schedule, where $V_{1,1}$ is sequenced on position n since the job $V_{1,1}$ belongs to E with $|\{E\}| = n$. Denote $P = \sum_{i \in \{V_{2,1}, V_{2,2}\} \bigcup \{\pi_1\} \bigcup \{V_{1,1}\}} p_i$.
  If $P - p_{V_{2,2}} \leq d$, then the schedule $\pi' = (V_{1,1}, V_{2,1}, \pi_1, V_{2,2}, j, \pi_2, V_{1,2})$ is also optimal, and the job $V_{1,1}$ is the first job.
  Otherwise, if $P - p_{V_{2,2}} > d$, then $T_j(\pi) > p_{V_{2,2}}$ and $T_{V_{1,1}}(\pi) > p_{V_{2,2}}$. Let us consider the schedule $\pi' = (V_{2,1}, \pi_1, V_{1,1}, j, V_{2,2}, \pi_2, V_{1,2})$. It is easy to show that $C_{V_{2,2}}(\pi') = C_j(\pi) < d + 2p_{V_{2,2}}$. Thus, we have

$$\sum_{j=1}^{2n+1} T_j(\pi) - \sum_{j=1}^{2n+1} T_j(\pi') > 0.$$

  Now it is easy to show that for the schedule $\pi'' = (V_{1,1}, \pi_1, V_{2,1}, j, V_{2,2}, \pi_2, V_{1,2})$, we have

$$\sum_{j=1}^{2n+1} T_j(\pi'') = \sum_{j=1}^{2n+1} T_j(\pi')$$

  since $\sum_{i \in \{V_{2,1}\} \bigcup \{\pi_1\} \bigcup \{V_{1,1}\}} p_i - d < p_{V_{2,1}}$, and so the job $V_{1,1}$ is the first job.
  (3.2) If $\{V_{2,1}, V_{2,2}\} \not\subset E$, then all jobs from sub-sequence E are not tardy in any schedule of the type $\pi = (E, F)$, where $V_{1,2} \in F$. Thus, the job $V_{1,1}$ is the first job as well.
(4) Now we consider only optimal schedules of the type $\pi = (V_{1,1}, \pi_1, \pi_2, V_{1,2})$. We use the same three steps (1)–(3) to prove that there exists an optimal schedule of the type $\pi = (V_{1,1}, V_{2,1}, \pi'_1, \pi'_2, V_{2,2}, V_{1,2})$.
(5) Then, by induction, we assume that there exists an optimal schedule of the type $\pi = (V_{1,1}, V_{2,1}, \ldots, V_{i-1,1}, \pi_1, \pi_2, V_{i-1,2}, \ldots, V_{2,2}, V_{1,2})$, $i \geq 3$, and we prove according to steps (1)–(3) that there exists an optimal schedule

$$\pi = (V_{1,1}, V_{2,1}, \ldots, V_{i-1,1}, V_{i,1}, \pi'_1,$$
$$\pi'_2, V_{i,2}, V_{i-1,2}, \ldots, V_{2,2}, V_{1,2}).$$

Thus, the theorem has been proven. □

We note that in a canonical schedule, there are either $n+1$ or $n+2$ tardy jobs (job $V_{n,1}$ can be tardy or on-time). Moreover, as we prove in the following theorem, in an optimal canonical schedule, there are only $n+1$ tardy jobs and thus, all jobs in sub-sequence E are on-time.

Theorem 6. *The instance of the partition problem has an answer "YES" if and only if in an optimal canonical schedule, the equality*

$$\sum_{j \in E} p_j = d$$

holds.

Proof. For a canonical schedule π, we have

$$\sum_{j=1}^{2n+1} T_j(\pi) = T_{V_{n,1}} + T_{2n+1} + \sum_{i=1}^{n} T_{V_{i,2}}$$

$$= T_{V_{n,1}} + T_{2n+1} + \sum_{i=1}^{n} (t_{n-i+2} + p_{2i} + \phi(i)b_i),$$

where

$$\phi(i) = \begin{cases} 1, & \text{if } V_{i,2} = V_{2i-1}, \\ 0, & \text{if } V_{i,2} = V_{2i}. \end{cases}$$

In addition, inequalities $0 \le T_{V_{n,1}} \le \frac{1}{2}\sum b_i$ and $1 \le T_{V_{2n+1}} \le \frac{1}{2}\sum b_i + 1$ hold. In fact, we want to minimize the value

$$T_{V_{n,1}} + T_{2n+1} + \sum_{i=1}^{n} \phi(i)b_i.$$

We attain the minimal value

$$T_{V_{n,1}} + T_{2n+1} + \sum_{i=1}^{n} \phi(i)b_i = 0 + 1 + \frac{1}{2}\sum_{i=1}^{n} b_i$$

if and only if $C_{V_{n,1}} = d$, i.e.,

$$\sum_{j \in E} p_j = d. \quad \square$$

Thus, the special case $1|NR, d_j = d, g_j = g| \sum T_j$ is *NP*-hard.

3.3. Properties of the problem $1|NR| \sum T_j$

We complete this section with some comments about the problem of minimizing total tardiness with arbitrary processing times. For this problem $1|NR| \sum T_j$, Lawler's proposition (Lawler, 1977)

$$\min(C_j^*, d_j) \le d_j' \le \max(C_j^*, d_j)$$

holds. However, the well-known elimination rule by Emmons (Szwarc et al., 1999), i.e.,

$$p_j > p_i, d_j \ge d_i \quad \Rightarrow \quad (i \to j)$$

does not hold. Here, the notation $(i \to j)$ means that there exists an optimal schedule in which the processing of job i precedes the processing of job j.

Let us consider an instance with $n = 3$ jobs and $p_1 = p > 1, p_2 = p_3 = 1, d_1 = d_2 = d_3 = p + 2, g_1 = 3, g_2 = g_3 = 2$, and $G(0) = 3, G(p) = 4$. For all optimal schedules of this instance, we have $(j \to i), p_j > p_i, d_j \ge d_i$. Therefore, we cannot use the exact pseudo-polynomial algorithm by Lawler (1977) for the problem $1 \| \sum T_j$ to solve the problem $1|NR| \sum T_j$.

4. Budget scheduling problems with makespan minimization

A *budget scheduling* problem is a financial scheduling problem described in this paper, where instead of the values g_j, values $g_j^- \ge 0$ and $g_j^+ \ge 0$ are given. The value g_j^- has the same meaning as g_j in the financial scheduling problem. However, at the completion time of job j, one has additional earnings g_j^+ of the resource.

If we have $g_j^- \ge g_j^+$ for all $j = 1, 2, \ldots, n$, then the new instance with $g_j = g_j^- - g_j^+$ is not equivalent to the original one. Let $G = \sum_{j=1}^{n}(g_j^- - g_j^+)$. If $\sum_{\forall t} G(t) < G + \max g_j^-$, then not all sequences (schedules) π are feasible. For example, let $n = 2$ and $g_1^- = 100, g_1^+ = 3, g_2^- = 3, g_2^+ = 2$ and $\sum_{\forall t} G(t) = 100$. Then schedule $(1, 2)$ is feasible but schedule $(2, 1)$ is not feasible.

We denote this problem as $1|NR, g_j^-, g_j^+|C_{\max}$. It is obvious that this problem is *NP*-hard in the strong sense (since the financial scheduling problem is a special case of the budget scheduling problem).

Remark 2. If there exists a feasible schedule for an instance of problem $1|NR, g_j^-, g_j^+, g_j^- > g_j^+|C_{\max}$, then the schedule $\pi = (1, 2, \ldots, n)$ with $g_1^+ \ge g_2^+ \ge \cdots \ge g_n^+$ is feasible as well.

If inequality $g_j^- > g_j^+$ does not hold for all $j = 1, 2, \ldots, n$, then we can use the following list scheduling algorithm for constructing a feasible schedule.

Algorithm A. First, all jobs $j \in N$ with $g_j^+ - g_j^- \ge 0$ are scheduled. In particular, schedule among these jobs the job with the minimal value g_j^-, if there is more than one job with this property, select the job with the largest value $g_j^+ - g_j^-$. If all jobs j with $g_j^+ - g_j^- \ge 0$ have been sequenced, schedule the remaining jobs according to non-increasing values g_i^+.

Moreover, we can give the following remark concerning the possibility of approximation of the problem under consideration.

Lemma 5. *The problem* $1|NR, g_j^-, g_j^+|C_{\max}$ *is in APX.*

Proof. Denote the set of times of earnings of the resource by $\{t_1, t_2, \ldots, t_{l-1}, t_l, \ldots, t_k\}$. Assume that for the times $\{t_1, t_2, \ldots, t_{l-1}, t_l\}$ of earnings of the resource, there already exists a feasible schedule, but for the set $\{t_1, t_2, \ldots, t_{l-1}\}$, there is no feasible schedule. Then t_l is a lower bound for the optimal objective function value $C_{\max}^*$. In addition, $\sum_{j=1}^{n} p_j$ is also a lower bound for $C_{\max}^*$. We can construct a feasible job sequence (schedule) π using Algorithm A, where all jobs are processed from time t_l without idle times. Thus, we obtain

$$C_{\max}(\pi) = t_l + \sum_{j=1}^{n} p_j \le 2C_{\max}^*. \quad \square$$

5. Concluding remarks

In this paper, we have considered single machine financial and budget scheduling problems. For the first class of problems with a non-renewable resource, we presented various complexity results. Then we considered several special cases of the problem of minimizing total tardiness. In addition, we presented some properties of feasible solutions for the budget scheduling problem of minimizing the makespan.

For future research, it is interesting to derive several elimination rules and properties of optimal solutions for the above mentioned single machine problems since most of the problems considered are *NP*-hard in the strong sense and thus the construction of pseudo-polynomial algorithms seems to be impossible. Another direction of future research is the consideration of shop and parallel machine problems with a non-renewable resource.

Acknowledgment

References

Carlier, J., Rinnooy Kan, A.H.G., 1982. Scheduling subject to nonrenewable-resource constraints. Oper. Res. Lett. 1 (2), 52–55.

Du, J., Leung, J.Y.-T., 1990. Minimizing total tardiness on one processor is *NP*-hard. Math. Oper. Res. 15, 483–495.

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

Janiak, A., 1986. One-machine scheduling problems with resource constraints. In: System Modelling and Optimization. Springer, Berlin, Heidelberg, pp. 358–364.

Janiak, A., Potts, C.N., Tautenhahn, T., 2000. Single maschine scheduling with nonlinear resource dependencies of release times. In: Abstract 14th Workshop on Discrete Optimization. Holzhau, Germany, May 2000.

Lawler, E.L., 1977. A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness. Ann. Discrete Math. 1, 331–342.

Lazarev, A.A., Gafarov, E.R., 2006. Special case of the single-machine total tardiness problem is *NP*-hard. J. Comput. System Sci. Int. 45 (3), 450–458.

Lazarev, A.A., Werner, F., 2009. Algorithms for special cases of the single machine total tardiness problem and an application to the even–odd partition problem. Math. Comput. Modelling 49 (9–10), 2061–2072.

Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P., 1977. Complexity of machine scheduling problems. Ann. Discrete Math. 1, 343–362.

Potts, C.N., Van Wassenhove, L.N., 1982. A decomposition algorithm for the single machine total tardiness problem. Oper. Res. Lett. 1, 363–377.

Slowinski, R., 1984. Preemptive scheduling of independent jobs on parallel machines subject to financial constraints. European J. Oper. Res. 15, 366–373.

Szwarc, W., Della Croce, F., Grosso, A., 1999. Solution of the single machine total tardiness problem. J. Sched. 2, 55–71.

Toker, A., Kondakci, S., Erkip, N., 1991. Scheduling under a non-renewable resource constraint. J. Oper. Res. Soc. 42 (9), 811–814.