Scheduling Problems with Financial Resource Constraints

E. R. Gafarov*, A. A. Lazarev*, F. Werner†

*Institute of Control Sciences RAS, axel73@mail.ru, lazarev@ipu.ru
†Otto-von-Guericke-Universität Magdeburg, frank.werner@ovgu.de

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. In this paper, we deal with single machine scheduling problems with a non-renewable resource, such problems are also referred to as *financial scheduling* problems.

The research in the area of scheduling problems with a non-renewable resource is rather limited. In [1], 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 [2]. Toker et al. [3] have shown that the problem of minimizing the makespan with a unit supply of a resource at each time period is polynomially solvable. Janiak et al. [4, 5] 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 $N = \{1, 2, \dots, n\}$ 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, energy, etc.) and a set of times $\{t_0, t_1, \dots, t_y\}$, $t_0 = 0$, $t_0 < t_1 < \dots < t_y$, of earnings of the resource. At each time t_i, $i = 0, 1, \dots, y$, 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 $\sum\limits_{j=1}^{n} g_j = \sum\limits_{i=0}^{y} G(t_i)$.

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

Moreover, we will call the sequence π a schedule too since one can compute $S = (S_{j_1}, S_{j_2}, \dots, S_{j_n})$ in $O(n)$ time 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.

1 Some Complexity Results

Theorem 1 *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 in the ordinary sense.*

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.

2 Problem $1|NR|\sum T_j$

For $1|NR, p_j = p|\sum T_j$, 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, $i = 1, 2, \ldots, y$, are processed in EDD order.

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

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.

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 $\alpha_t = 1$ [3]. 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 $1||\sum T_j$. Given an instance of the problem $1||\sum T_j$ 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 [8] since we obtain a problem $1||\sum T_j$ with processing times g_j.

According to the definition in [7], we have $LENGTH[I] = n + y$. In fact, the string x consists of $2n + 2y + 3$ numbers. However, if we consider problem $1|NR : \alpha_t = 1, p_j = 0|\sum T_j$ as a special case of 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, 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., $LENGTH[I] = n$.

Let us now 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.

Lemma 1 *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.

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

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

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

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

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.*

Special Case $1|NR, d_j = d, g_j = g|\sum T_j$

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 $N = \{1, 2, \ldots, 2n+1\}$:

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

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 the 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.*

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.

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

3 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 $g_j^- \geq 0$ and $g_j^+ \geq 0$ 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 $g_j^- \geq g_j^+$ 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.

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).

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 $g_1^+ \geq g_2^+ \geq \ldots \geq g_n^+$ 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 $g_j^+ - g_j^- \geq 0$ are scheduled. In particular, schedule among these jobs the job with the earliest possible starting time, 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 $g_j^+ - g_j^- \geq 0$ have been sequenced, schedule the remaining jobs according to non-increasing values g_i^+.

Lemma 2 *The problem $1|NR, g_j^-, g_j^+|C_{max}$ is in APX.*

References

[1] J. Carlier and A.H.G. Rinnooy Kan, *Scheduling subject to Nonrenewable-Resource Constraints.* Oper. Res. Lett. 1(2), 52 – 55, 1982.

[2] R. Slowinski, *Preemptive Scheduling of Independent Jobs on Parallel Machines subject to Financial Constraints.* European J. Oper. Res. 15, 366 – 373, 1984.

[3] A. Toker, S. Kondakci and N. Erkip, *Scheduling Under a Non-renewable Resource Constraint.* J. Oper. Res. Soc. 42(9), 811 – 814, 1991.

[4] A. Janiak, *One-Machine Scheduling Problems with Resource Constraints.* System Modelling and Optimization. Springer Berlin / Heidelberg, 358 – 364, 1986.

[5] A. Janiak, C.N. Potts and T. Tautenhahn, *Single Maschine Scheduling with Nonlinear Resource Dependencies of Release Times.* Abstract 14th Workshop on Discrete Optimization, Holzhau/Germany, May 2000.

[6] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, *Complexity of Machine Scheduling Problems.* Annals Discrete Math. 1, 343 – 362, 1977.

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

[8] E.L. Lawler, *A Pseudopolynomial Algorithm for Sequencing Jobs to Minimize Total Tardiness.* Ann. Discrete Math. 1, 331 – 342, 1977.

[9] C.N. Potts and L.N. Van Wassenhove, *A Decomposition Algorithm for the Single Machine Total Tardiness Problem.* Oper. Res. Lett. 1, 363 – 377, 1982.