ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: AUTOMATYKA z. 123 Nr kol. 1389

Aleksander BACHMAN, Adam JANIAK
Politechnika Wrocławska

MINIMIZING THE MAXIMUM COMPLETION TIME FOR THE DETERIORATING JOBS WITH READY TIMES*

Summary. In this paper we consider the single machine problem with the maximum completion time criterion. Job processing time is described by a nondecreasing, linear function dependent on the job processing start time. Job ready time is also given for each job. We assume, that for each job, its processing time deterioration begins at its ready time. Each job is available for the time not smaller than its ready time. The job processing time consists of two parts: one is constant and the other one: variable, start time dependent. The variable part is characterised by the growth rate, which describes how fast the job processing time deteriorate. If the job begins exactly at its ready time, its processing time is equal only to its constant part. In this paper, for the problem mentioned above, we present the NP-completeness proof. We also present two heuristic algorithms. The first heuristic algorithm bases on the adjacent jobs interchanging and the second one on the extended Jackson's rule. In this paper we compare presented algorithms basing on the computational results.

MINIMALIZACJA CZASU ZAKOŃCZENIA WYKONYWANIA ZADAŃ CZASOWO ZALEŻNYCH PRZY ZADANYCH TERMINACH DOSTĘPNOŚCI

Streszczenie. W pracy rozpatrywany jest jednomaszynowy problem minimalizacji czasu zakończenia wykonywania zadań czasowo zależnych przy zadanych terminach dostępności. Założono, że wydłużenie czasu wykonywania zadania, opisanego funkcją liniową, następuje od momentu jego dostępności. Zostało pokazane, że powyższy problem jest NP-zupełny. Przedstawiono dwa algorytmy rozwiązujące rozpatrywany problem w sposób przybliżony.

1. Introduction

There are many real scheduling environments, where the job processing time is start time dependent. Some notable applications can be found in the firefighting, metallurgy or military. Kunnathur and Gupta [4] mentioned the problem of scheduling resources when there are several fires to be controlled, where the time and effort required to control a fire increases if

* The paper was partially supported by the KBN Grant No 8T11F001 11

there is a delay in the start of the fire fighting effort. They also discussed a specific case of steel rolling mill. The time needed to heat an ingot depends on its size and its current temperature. The temperature of an ingot, before it can start rolling, depends on the amount of time the ingot has been waiting in the queue to be executed after it has been taken out of the furnace. Ho, Leung and Wei [3] presented the military application, where the job consists of destroying an aerial threat and its execution time decreases with time as the threat gets closer. According to the application, there are many different models describing the phenomenon of processing time deterioration. Most of them are the combinations of the linear functions.

We consider the problem, where the job processing time depends on the execution starting time and the ready time as well. The job processing time is given as a nondecreasing linear function containing constant and variable parts. The constant part represents the minimum requirements to accomplish the job's execution if its starting moment is equal to its ready time. The variable part, characterised by the growth rate, describes how fast the job processing time deteriorate. We consider the single machine scheduling problem with the maximum completion time minimization.

In the second section we formulate the problem more precisely. The third section contains NP-completeness results. In the fourth section we presented two heuristic algorithms, which have been compared in the fifth section. The last section contains some final remarks.

2. Problem formulation

A set of n independent, single operation and not preemptable jobs is given. Let $J = \{1,2,\ldots,n\}$ denotes a set of job indices. The job processing time p_j is given as a nondecreasing, linear function dependent on its start and ready times, which are denoted as S_j and r_j respectively, and is described as follows:

$$p_j(S_j) = a_j + b_j(S_j - r_j);\ \ a_j > 0,\ \ b_j > 0,\ \ S_j \geq r_j, \tag{1}$$

where a_j and b_j are the constant part and the growth rate of the job processing time respectively. We assume that for each job its deterioration begins, when its start time is greater than its ready time. We introduce also a constant $C_0 = 0$, which denotes the completion time of the zero job (time, when machine starts its duty). It is easy to proof using mathematical induction that for such a formulated problem (for the jobs with ready times equal to 0 and

common growth rates) the completion time of the job placed on the ith position in any schedule is given as follows:

$$C_{[i]} = \sum_{j=1}^{i} a_{[j]}(b+1)^{i-j} + C_0(b+1)^i; \qquad b = b_{[i]};\ r_{[i]} = 0,\ i = 1,\ldots,n. \tag{2}$$

3. NP-completeness results

Now we show that the decision form of the TDPT problem is NP-complete in the strong sense by reducing to it strongly NP-complete 3-Partition problem [3]. The decision forms for the 3-Partition problem and the TDPT problem are defined as follows:

TDPT: given a single machine problem with uninterrupted job processing times given as (1) with the parameters a_i, b_i, r_i, $i = 1,\ldots,n$, and number y, where n is a positive integer, a_i, b_i, y are positive rational and r_i are nonnegative rational; does there exist such a schedule, where the maximum completion time is not greater than y?

3-Partition: given a set $X = (x_1,\ldots,x_{3m})$ of $3m$ integers such that $\sum_{i=1}^{3m} x_i = mB$ and $\frac{B}{4} < x_i < \frac{B}{2}$ for each $i = 1,\ldots,3m$; can X be partitioned into $X_1, X_2,\ldots,X_m$ such that $\sum_{x_i \in X_j} x_i = B$ for each $j = 1,\ldots,m$?

Now we give a reduction from the 3-Partition problem to the TDPT problem. Given an instance $X = (x_1,\ldots,x_{3m})$ of the 3-Partition problem, we construct an instance of the TDPT problem. TDPT problem consists of $n=4m-1$ jobs, where:

$$r_i = \begin{cases} iB + 2i - 1 - \dfrac{1}{4m^3(B+2)}, & i = 1,\ldots,m-1, \\ 0, & i = m,\ldots,4m-1, \end{cases}$$

$$a_i = \begin{cases} 1, & i = 1,\ldots,m-1, \\ x_{i-m+1}, & i = m,\ldots,4m-1, \end{cases} \qquad b_i = \begin{cases} 8m^3(B+2), & i = 1,\ldots,m-1, \\ \dfrac{1}{2m^3(B+2)}, & i = m,\ldots,4m-1. \end{cases}$$

The number y is equal to $mB+2m-1$.

Above construction can be done in O(n) time. In our further consideration we will call the first $m-1$ jobs the enforcer jobs and the last $3m$ jobs the partition jobs.

Now we show that the TDPT problem has a solution if and only if the 3-Partition problem has a solution.

Lemma 1. If the instance of the 3-Partition problem has a solution, the constructed instance of TDPT problem has also a solution.

Proof. Let $X_1, X_2, \ldots, X_m$ be a solution of the 3-Partition problem, hence $\sum_{x_i \in X_j} x_i = B$ for $j = 1, \ldots, m$. The schedule, which solves the TDPT problem is characterised as follows. The each one of the enforcer jobs starts its execution at its ready time, hence the processing time of each of the enforcer job is equal to one. The time interval [*0, ..., mB+2m-1*] is in this way partitioned by the enforcer jobs into *m* intervals, $I_1, I_2, \ldots, I_m$, with the length of each interval exactly $B+1$. For each $x_i \in X_j$, schedule the jobs constructed on the base of the elements from the appropriate subset in the interval I_j. Jobs assigned to the same interval can be scheduled in an arbitrary order. Now we show that the maximum completion time of the jobs from two intervals: I_1 and I_m are not greater than $B+1-\frac{1}{4m^3(B+2)}$ and $mB+2m-1$ respectively. From (2) we have: $C_{X_1} = \sum_{j=1}^{3} a_{[1]}(b_{[1]}+1)^{3-j} < \left(\frac{1}{2m^3(B+2)}+1\right)^2 \sum_{j=1}^{3} x_{[1]} < \left(\frac{6}{2m^3(B+2)}+1\right)B < B+1-\frac{1}{4m^3(B+2)}$.

Since the maximum completion time of the partition jobs assigned to the first interval is smaller than the ready time of the first enforcer job, that job can start its execution exactly at its ready time and complete it in $B+2-\frac{1}{4m^3(B+2)}$.

It is easy to show that for each interval I_j; $j = 1, \ldots, m-1$ the partition jobs assigned to I_j, complete their execution before the ready time of the enforcer job *j*, hence each of the enforcer jobs completes in $C_j = jB+2j-\frac{1}{4m^3(B+2)}$. The completion time of the last, *m-1* enforcer job is given by: $C_{m-1} = (m-1)B+2(m-1)-\frac{1}{4m^3(B+2)}$, what gives the start time of the jobs assigned to the last interval. The maximum completion time is then, on the base of (2), given as follows:

$$C_{X_m} = \sum_{j=4m-3}^{4m-1} a_{[1]}(b_{[1]}+1)^{4m-1-j} + C_{m-1}\left(\frac{1}{2m^3(B+2)}+1\right)^3 < \left(\frac{1}{2m^3(B+2)}+1\right)^3\left[\sum_{j=4m-3}^{4m-1} a_{[1]} + C_{m-1}\right] =$$

$$= \frac{3}{m^2} - \frac{6}{m^3(B+2)} - \frac{3}{4m^6(B+2)^2} + m(B+2) - 2 - \frac{1}{4m^3(B+2)} < m(B+2) - 1 - \frac{1}{4m^3(B+2)} < mB + 2m - 1.$$

The maximum completion time of the schedule is not greater than *mB+2m-1*, hence it has been shown that if the 3-Partition problem has a solution, the TDPT problem has also a solution. □

Now we have to show that if the TDPT problem has a solution the 3-Partition problem has also a solution, but basing on the mathematical logic, it is equivalent if we show that if the 3-Partition problem has no solution the TDPT problem has also no solution.

Lemma 2. If the given instance of the 3-Partition problem has no solution, the constructed instance of the TDPT problem also has no solution.

Proof. If the instance of the 3-Partition problem has no solution, there are at least two subsets, where the sum of elements is not equal, hence $\sum_{x_i \in X_j} x_i \neq \sum_{x_i \in X_k} x_i;\ j = 1,\ldots,m,\ k = 1,\ldots,m,\ j \neq k$.

Because the processing time deterioration of the enforcer jobs grows very much with the start time greater than its ready time, it is enough when we consider two following cases:

i) $\sum_{x_i \in X_j} x_i = B+1$, and $\sum_{x_i \in X_k} x_i = B-1$ and ii) $\sum_{x_i \in X_j} x_i = B-1$, and $\sum_{x_i \in X_k} x_i = B+1$.

Let's consider the first case. Without loosing the generality, we can assume that in the following three subsets the sum of elements is given by: $\sum_{x_i \in X_1} x_i = B$; $\sum_{x_i \in X_2} x_i = B+1$; $\sum_{x_i \in X_3} x_i = B-1$. The minimal, positive job processing time deterioration is equal to $\frac{1}{2m^3(B+2)}$.

Since the sum of job processing time constant parts constructed on the base of the elements from the subset X_1 is equal to B, the completion time of the interval I_1 is not greater than the ready time of the first enforcer job. Hence the partition jobs constructed on the base of the elements from the second subset can start their processing at $C_1 = B+2-\frac{1}{4m^3(B+2)}$. The maximum completion time of the second interval is then given as follows: $C_{x_1} > \sum_{j=4}^{6} a_{(j)} + C_1 + \frac{1}{2m^3(B+2)} > 2B+3+\frac{1}{4m^3(B+2)} > 2B+3-\frac{1}{4m^3(B+2)}$. The second enforcer job has then the late start and completes at least in $C_2 = 2B+8+\frac{1}{4m^3(B+2)}$. The difference between the ready time of the third enforcer job and the completion time of the second enforcer job is given by: $r_3 - C_2 = 3B+5-\frac{1}{4m^3(B+2)} - 2B-8-\frac{1}{4m^3(B+2)} = B-3-\frac{1}{2m^3(B+2)} < B-1+\frac{1}{2m^3(B+2)}$, and this means that the maximum completion time of the partition jobs assigned to the third interval is greater than the ready time of the third enforcer job, hence the processing time of the third enforcer job will be greater than 1. If in the other subsets the sum of job processing time constant parts is equal to B, each of the other enforcer jobs will start its execution after its ready time, hence the schedule maximum completion time will be greater than $mB+2m-1$.

Now we consider the second case. If in any subset the sum of job processing time constant parts is smaller than B, there must exist a subset, where the sum of job processing

time constant parts is greater than B, because $\sum_{i=1}^{3m} x_i = mB$. Hence basing on the previous case we can show that the schedule maximum completion time will be greater than $mB+2m-1$.

We showed that if the 3-Partition problem has no solution the TDPT problem has also no solution. □

Lemmas 1 and 2 immediately lead to the following theorem.

Theorem 1. The TDPT problem is NP-complete in the strong sense for a set of jobs with arbitrary ready times.

Using the above strategy, we can show that the decision form of the single machine problem with time dependent processing times (TDPT) and with two distinct ready times is NP-complete in the ordinary sense, by reducing to it NP-complete Partition Problem [7]. The Partition Problem is defined as follows:

Partition: given positive integers $X=(x_1,\ldots,x_m)$; does there exist a subset $Y \subset X=(x_1,\ldots,x_m)$ such that $\sum_{i\in Y} x_i = \sum_{i\in X\setminus Y} x_i = B$?

The reduction from the Partition Problem to the TDPT problem is described in the following way:

$$r_i=\begin{cases}0, & i=1,\ldots,m,\\ B+1-\dfrac{1}{4m^3(B+1)}, & i=m+1,\end{cases} \quad a_i=\begin{cases}x_i, & i=1,\ldots,m,\\ 1, & i=m+1,\end{cases} \quad b_i=\begin{cases}\dfrac{1}{2m^2(B+1)}, & i=1,\ldots,m,\\ 8m^2(B+1), & i=m+1.\end{cases}$$

The number y is equal to $2B+3$.

Above construction can be done in O(n) time.

Now using the same proof technique as before, we have immediately the following result.

Theorem 2. The TDPT problem is NP-complete in ordinary sense for a set of jobs with two distinct ready times.

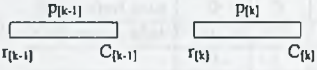
4. Heuristic algorithms

In this section we present two heuristic algorithms, which base on the adjacent jobs interchanging. First we show some properties of the problem and then, using them we formulate the steps of the algorithms.

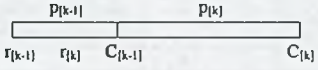
Definition 1. There are given two permutations of the jobs: π and π'. The permutation π' is achieved from the permutation π by interchanging the jobs from the kth and $k+1$st positions.

The relation between the completion time and the ready time of two adjacent jobs placed in any permutation gives two possibilities of the way the machine is working:

a) with idle times; $C_{[k-1]} < r_{[k]}$



b) without idle times; $C_{[k-1]} \geq r_{[k]}$



To join this two possibilities we introduce a variable Δ, which describes the machine idle time between proceeding the jobs. This variable is related to the position of the job placed in a permutation and is defined as follows:

Definition 2. The machine idle time is always nonnegative and is given as:

$$\Delta_{[k]} = \max\left\{0, r_{[k]} - C_{[k-1]}\right\}. \tag{3}$$

Basing on (3) and the following $a'_{[k]} = a_{[k+1]}$, $a'_{[k+1]} = a_{[k]}$, $b'_{[k]} = b_{[k+1]}$, $b'_{[k+1]} = b_{[k]}$, $r'_{[k]} = r_{[k+1]}$, $r'_{[k+1]} = r_{[k]}$, we can find the expressions of the completion times for the jobs placed on the kth and the $k+1$st positions in the permutations π and π':

$$C_{[k]} = a_{[k]} + (b_{[k]} + 1)(C_{[k-1]} + \Delta_{[k]}) - b_{[k]}r_{[k]},$$

$$C_{[k+1]} = a_{[k+1]} + (b_{[k+1]} + 1)a_{[k]} + (b_{[k+1]} + 1)(b_{[k]} + 1)C_{[k-1]} + \\ + (b_{[k+1]} + 1)(b_{[k]} + 1)\Delta_{[k]} - (b_{[k+1]} + 1)b_{[k]}r_{[k]} - b_{[k+1]}r_{[k+1]} + (b_{[k+1]} + 1)\Delta_{[k+1]},$$

$$C'_{[k]} = a_{[k+1]} + (b_{[k+1]} + 1)(C_{[k-1]} + \Delta'_{[k]}) - b_{[k+1]}r_{[k+1]},$$

$$C'_{[k+1]} = a_{[k]} + (b_{[k]} + 1)a_{[k+1]} + (b_{[k]} + 1)(b_{[k+1]} + 1)C_{[k-1]} + \\ + (b_{[k]} + 1)(b_{[k+1]} + 1)\Delta'_{[k]} - (b_{[k]} + 1)b_{[k+1]}r_{[k+1]} - b_{[k]}r_{[k]} + (b_{[k]} + 1)\Delta'_{[k+1]}.$$

If the difference between completion times of the jobs placed on appropriated positions in permutations π and π', given as $C_{[k+1]} - C'_{[k+1]}$, is positive then permutation π' is better than permutation π. It means that the interchanging jobs from the kth and the $k+1$st positions in permutation π is advantageous. We have:

$$C_{[k+1]} - C'_{[k+1]} = b_{[k+1]}a_{[k]} - b_{[k]}a_{[k+1]} + b_{[k]}b_{[k+1]}(r_{[k+1]} - r_{[k]}) + (b_{[k+1]} + 1)(b_{[k]} + 1)(\Delta_{[k]} - \Delta'_{[k]}) + \\ + (b_{[k+1]} + 1)\Delta_{[k+1]} - (b_{[k]} + 1)\Delta'_{[k+1]}. \tag{4}$$

To check the sign of the expression $C_{[k+1]} - C'_{[k+1]}$ we have to examine the cases, when variables Δ and Δ' are described more clearly.

Table 1

The existence of machine idle times before and after interchanging two adjacent jobs

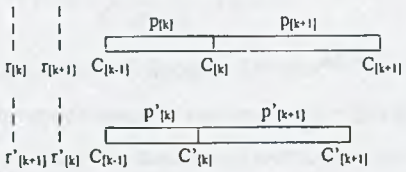
	$\Delta_{[k]}$	$\Delta_{[k+1]}$	$\Delta'_{[k]}$	$\Delta'_{[k+1]}$			$\Delta_{[k]}$	$\Delta_{[k+1]}$	$\Delta'_{[k]}$	$\Delta'_{[k+1]}$	
A	0	0	0	0		D	1	0	0	0	
	0	0	0	1	case impossible	E	1	0	0	1	
B	0	0	1	0		F	1	0	1	0	
	0	0	1	1	case impossible	G	1	0	1	1	
	0	1	0	0	case impossible		1	1	0	0	case impossible
	0	1	0	1	case impossible		1	1	0	1	case impossible
C	0	1	1	0		H	1	1	1	0	
	0	1	1	1	case impossible		1	1	1	1	case impossible

In above table „1" means existence and „0" non existence of positive value of Δ. There are sixteen cases, which describe the existence of machine idle times before and after the adjacent jobs interchanging placed on the kth and the $k+1$st position in the permutation π. In the impossible eight cases some conditions for ready times are contradictory.

Now we will find the final conditions for the expression (4) in each case. Since the size of the paper is limited we show only the final conditions in case A, consideration of the remaining cases can be done in the same way.

Case A

$\eta_{[k]} \le C_{[k-1]} \ \wedge \ \eta_{[k+1]} \le C_{[k-1]}$.



$$C_{[k+1]} - C'_{[k+1]} = b_{[k+1]}a_{[k]} - b_{[k]}a_{[k+1]} + b_{[k]}b_{[k+1]}(\eta_{[k+1]} - \eta_{[k]}),$$

$$C_{[k+1]} - C'_{[k+1]} \le 0,$$

$$b_{[k+1]}a_{[k]} - b_{[k]}a_{[k+1]} + b_{[k]}b_{[k+1]}(\eta_{[k+1]} - \eta_{[k]}) \le 0,$$

$$\frac{a_{[k]}}{b_{[k]}} - \eta_{[k]} \le \frac{a_{[k+1]}}{b_{[k+1]}} - \eta_{[k+1]}.$$

Conclusions

Analysis of eight possible cases leads us to the following corollaries:

- in four cases (C, E, G and H) better permutation can be achieved only if the jobs are scheduled in nondecreasing order of their ready times, hence $\eta_{[k]} \le \eta_{[k+1]}$, $k = 1..n-1$,
- in three cases (A, B, F) the jobs are initially placed in nondecreasing order of their ready times but if the appropriate conditions are fulfilled this order can be changed,

- case D is opposite to the case B; what means, that interchanging jobs in the case B excludes the possibility of interchanging jobs the case D; in case D the jobs are initially placed in nonincreasing order of their ready times.

If we assume, that jobs are initially scheduled in nondecreasing order of their ready times, we make sure, that for cases C, E, G and H jobs are in appropriate order. Only what we have to check is, if between every pair of adjacent jobs one of the cases A, B or F is fulfilled.

4.1. Heuristic Algorithm H_1

As we stated before in four cases, scheduling jobs in nondecreasing order of their ready times gives a better permutation. In three other cases such a scheduling is initially required. Basing on this corollaries we achieve the first step of the algorithm: **schedule the jobs in nondecreasing order of their ready times**. Then we showed, that the advantageous jobs interchanging can be obtained only in three cases: A, B, F. To check out if the jobs interchanging condition is fulfilled, first we have to find out if the jobs are in the appropriate case. This statement leads us to the second step of the algorithm: **for every pair of adjacent jobs find out if the case A, B or F proceeds and then if the appropriate condition is fulfilled interchange these adjacent jobs**. In the following two tables we placed the necessary conditions for proceeding cases and the sufficient conditions for the adjacent jobs interchanging.

Table 2

The necessary conditions for proceeding cases

case	case conditions
A	$r_{[k]} \le C_{[k-1]}$ $r_{[k+1]} \le C_{[k-1]}$
B	$r_{[k]} \le C_{[k-1]}$ $C_{[k-1]} < r_{[k+1]}$ $r_{[k+1]} \le a_{[k]} + (b_{[k]} + 1)C_{[k-1]} - b_{[k]}r_{[k]}$
F	$C_{[k-1]} < r_{[k]}$ $r_{[k]} < r_{[k+1]} + a_{[k+1]}$, *always fulfilled* $C_{[k-1]} < r_{[k+1]}$ $r_{[k+1]} < r_{[k]} + a_{[k]}$

Table 3

The sufficient conditions for the adjacent jobs interchanging

case	jobs interchanging condition
A	$\frac{a_{[k]}}{b_{[k]}} - r_{[k]} > \frac{a_{[k+1]}}{b_{[k+1]}} - r_{[k+1]}$
B	$(C_{[k-1]} - r_{[k+1]})(b_{[k]} + b_{[k+1]} + 1) +$ $+ b_{[k+1]}a_{[k]} - b_{[k]}a_{[k+1]} +$ $+ b_{[k]}b_{[k+1]}(C_{[k-1]} - r_{[k]}) > 0$
F	$b_{[k+1]}a_{[k]} - b_{[k]}a_{[k+1]} +$ $+ (b_{[k]} + b_{[k+1]} + 1)(r_{[k]} - r_{[k+1]}) > 0$

We assume that interchanging jobs will be proceed in ascending order of the places the jobs take in the permutation. The jobs interchanging, which were in case B or F gives only two possibilities for the jobs from the previous positions. In both achieved cases (C and H) interchanging jobs is always disadvantageous.

Interchanging jobs, which were in case A can give three cases for the previous placed jobs: B, F or A. To set the jobs, which are in the case B or F, in appropriate order we need only one step, but if in the effect of the jobs interchanging from the *k*th and *k+1*st positions we achieve the case A again, we will have to, in the worst case, repeat *k-1* times the second step of the algorithm. That statement gives us two following corollaries:

- complexity of the second step of the algorithm is $O(n^2)$,
- the stop condition is bounded with the existence of the case A; if by the last whole permutation checking there was no case A, we have to check the whole permutation only once more.

4.2. Heuristic Algorithm H_2

Our second heuristic algorithm bases on the extended Jackson's rule [6], which can be formulated as follows: **from the set of the available jobs schedule first that one with the smallest value of the ratio** $\frac{a_i}{b_i} - r_i$ (see case A in the heuristic H_1). Above rule is the second step in our algorithm. In the first step we **place the jobs in nondecreasing order of their ready times.** The complexity of that heuristic is O(nlogn), if the queue of the available jobs has, for instance, the heap organisation.

5. Computational experiments

Three set of problems were examined: no machine idle times, some machine idle times and very much machine idle times. In each set, the parameters a_i and b_i were generated according to the uniform distribution from the intervals (0.0, 1.0] and (0.0, 0.5] respectively. The parameter r_i were generated according to the uniform distribution from three different intervals to achieve three different sets of problems: $(0, 2\sum_{i=1}^{n} a_i]$, $(0, 3\sum_{i=1}^{n} a_i]$ and $(0, 4\sum_{i=1}^{n} a_i]$. It was assumed that n = 100. The above heuristic algorithms were coded in C++ and run on the Pentium 75MHz with 16 MB RAM. In each set 100 randomly generated examples have been

tested. The arithmetic average of the 100 maximum completion time values for each set of generated examples are given in the following table:

Table 4

Average values given by two heuristic algorithms

r_i interval	H_1	H_2	H_1/H_2
$(0, 2\sum_{i=1}^{n} a_i]$	1,86 E+08	2,18 E+08	8,27 E-01
$(0, 3\sum_{i=1}^{n} a_i]$	6,86 E+06	7,16 E+06	9,77 E-01
$(0, 4\sum_{i=1}^{n} a_i]$	1,99 E+02	1,99 E+02	9,99 E-01

The above table shows that the Heuristic H_1 performs slightly better than Heuristic H_2. The average relative difference between the heuristics is less than 18%. For the problems with some and very much machine idle times both algorithms performed almost the same solutions. For the problems with no idle times Heuristic H_1 generated definitely better solutions than the Heuristic H_2.

6. Final conclusions

Since the size of the paper was limited, we had to cut off some estimations used in the proof of the strong NP-completeness. We couldn't also show all the cases considered in the Heuristic H_1. The figures showing the situation after the adjacent jobs interchanging in H_1 were also removed. We hope the simplifications don't make the paper difficult to understand.

REFERENCES

1. Alidaee B., Landram F., Scheduling deteriorating jobs on a single machine to minimize the maximum processing times, *International Journal of System Science*, 27/5 (1996), 507-510.
2. Browne S., Yechiali U., Scheduling deteriorating jobs on single processor, *Operations Research*, 38/3 (1990), 495-498.
3. Garey M.R., Johnson D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.
4. Gupta J.N.D., Gupta S.K., Single facility scheduling with nonlinear processing times, *Computers and Industrial Engineering*, 14/4 (1988), 387-393.
5. Ho K.I-J., Leung J.Y-T., Wei W-D., Complexity of scheduling tasks with time-dependent execution times, *Information Processing Letters*, 48 (1993), 315-320.

6. Jackson J.R., An extension of Johnson's results on job lot scheduling, *Naval Research Logistic Quart.*, 3, 1956, 201-203.
7. Kunnathur A.S., Gupta S.K., Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem, *European Journal of Operational Research*, 47 (1990), 56-64.
8. Mosheiov G., V-shaped policies to schedule deteriorating jobs, *Operations Research*, 39/6 (1991), 979-991.

Recenzent: Prof.dr hab.inż. Jan Węglarz

Streszczenie

W pracy rozpatrywany jest jednomaszynowy problem szeregowania zadań przy kryterium minimalizacji czasu zakończenia wykonywania wszystkich zadań. Czas wykonywania zadania jest opisany niemalejącą funkcją liniową, zależną od momentu rozpoczęcia wykonywania i zadanego terminu gotowości. Zależność opisująca zmianę czasu wykonywania zadania składa się z dwóch części: stałej i zmiennej. Część stała odpowiada minimalnym wymaganiom technologicznym na wykonanie zadania. Część zmienna, charakteryzowana przez współczynnik wzrostu, opisuje sposób w jaki zadanie ulega wydłużeniu. W pracy pokazano, że powyżej opisany problem szeregowania jest NP-zupełny. Przedstawiono ponadto dwa algorytmy rozwiązujące ten problem w sposób przybliżony.