Faculty of Mathematical Sciences

University of Twente
The Netherlands

P.O. Box 217
7500 AE Enschede
The Netherlands
Phone: +31-53-4893400
Fax: +31-53-4893114
Email: memo@math.utwente.nl
www.math.utwente.nl/publications

MEMORANDUM No. 1605

How useful are preemptive schedules?

P. BRUCKER¹, S. HEITMANN¹ AND J.L. HURINK

DECEMBER 2001

ISSN 0169-2690

¹Department of Mathematics/Informatics, University of Osnabrück

How Useful are Preemptive Schedules?

Peter Brucker

Silvia Heitmann

Department of Mathematics/Informatics

University of Osnabrück

Johann Hurink

Faculty of Mathematical Sciences

University of Twente

November 2001

Abstract

Machine scheduling admits two options to process jobs. In a preemptive mode processing may be interrupted and resumed later even on a different machine. In a nonpreemptive mode interruptions are not allowed. Usually, the possibility to preempt jobs leads to better performance values. However, also examples exist where preemptions do not improve the performance. This paper gives an overview of existing and new results on this topic for single and parallel machine scheduling problems.

Keywords: Preemption, parallel machine scheduling.

Subject classification: 90B35

1 Introduction

Machine scheduling admits two options to process jobs. In a **preemptive** mode processing may be interrupted and resumed later even on a different machine. In a **nonpreemptive** mode interruptions are not allowed, i.e. a job has to be processed continuously on the same machine. Clearly one has more degrees of freedom when jobs can be scheduled preemptively. Often the objective value of an optimal schedule improves when switching to the preemptive mode. However, preemption may also be **redundant**, i.e. the objective value may not be improved by allowing preemptions.

The aim of this paper is to identify parallel machine scheduling problems for which preemption is redundant. In the presentation the well known $\alpha \mid \beta \mid \gamma$ -notation of Graham et al. [6] will be used. Furthermore, all data are assumed to be integers.

For single machine problems, which may be considered as a special case, one can completely classify redundant and nonredundant problems, if the objective functions are regular: If all jobs are available at the same time (no release dates) redundancy can be proved by simple exchange arguments. The situation is slightly more complex when release times are present. It can be shown that preemption is redundant for problems with unit processing times and for the makespan problem $1 \mid r_j \mid C_{\max}$. On the other hand preemptions may provide better solutions for $1 \mid r_j \mid \sum C_j$ and $1 \mid r_j \mid L_{\max}$ as the example in Fig. 1 shows. All other commonly used objective functions can be considered as generalizations of these examples. Thus, preemptions are nonredundant for these objectives as well.

$$\begin{array}{lll} p_1 = 4 & r_1 = 0 & d_1 = 5 \\ p_2 = 1 & r_2 = 1 & d_2 = 2 \end{array}$$

Preemptive Schedule
 $L_{\max} = 0 \quad \sum C_j = 7$

Optimal nonpreemptive Schedule
 $L_{\max} = 1 \quad \sum C_j = 8$

Figure 1: Instance for $1 \mid r_j \mid L_{\max}$ and $1 \mid r_j \mid \sum C_j$ where preemptions are nonredundant

For parallel machines the situation is quite different due to the fact that after preemption a job can be resumed on a different machine. For this reason preemption is nonredundant already for $P2 \mid p = 2 \mid C_{\max}$ (see Fig. 2) and, thus, for all regular objective functions based on due dates. On the other hand a classical result of McNaughton [8] shows that for $P \parallel \sum w_j C_j$ preemption is redundant. This result was strengthened by Du et al. [5] to problems in which chain precedence constraints are added ($P \mid chains \mid \sum w_j C_j$). If one restricts to unit processing times further results are available. Recently, Baptiste & Timkovsky [1] showed that preemptions are redundant for $P2 \mid p_j = 1; r_j; outtree \mid \sum C_j$. This result has been generalized to an arbitrary number of machines by Brucker et al. [2]. On the other hand, Baptiste & Timkovsky [1] showed that preemption is nonredundant for $P2 \mid p_j = 1; outtree \mid \sum w_j C_j$ and $P2 \mid p_j = 1; intree \mid \sum C_j$.

In this paper we show that for the following problems with unit processing times preemption is redundant: $P \mid p_j = 1; r_j \mid \sum w_j U_j$ and $P \mid p_j = 1; intree \mid \sum w_j U_j$

Figure 2: Preemption is nonredundant for $P2 \mid p_j = 2 \mid C_{\max}$

(Section 2), $P \mid p_j = 1 \mid \sum T_j$ (Section 3), and $P2 \mid p_j = 1; r_j \mid \sum w_j T_j$ (Section 4). Thus, the complexity status of the preemptive versions of these problems is the same as for the nonpreemptive versions which provides some new complexity results. We close the paper with some concluding remarks.

2 Problems $P \mid p_j = 1; r_j \mid \sum w_j U_j$ and $P \mid p_j = 1; \text{intree} \mid \sum w_j U_j$

In this section, we consider problems with the weighted sum of late jobs as objective function. It is shown that if release times or intree-precedences are given, preemptions do not lead to better schedules.

We first consider the time window problem $P \mid p_j = 1; r_j; d_j \mid -$, where for each job j a time window $[r_j, d_j]$ is given by its release time and its due date. This problem has a feasible solution iff a schedule exists in which each job is processed in its time window. After showing that the time window problem has a feasible solution iff the corresponding preemptive problem has a feasible solution, we can use this result in order to derive that preemptions are redundant for problem $P \mid p_j = 1; r_j \mid \sum w_j U_j$.

Lemma 1 : The time window problem $P \mid p_j = 1; r_j; d_j \mid -$ has a feasible solution if and only if the corresponding preemptive problem has a feasible solution.

Proof: The preemptive problem $P \mid p_j = 1; r_j; d_j; pmtn \mid -$ is equivalent to the following transshipment problem with upper capacities u_{jk} on the arcs (j, k) .

In this network node k corresponds with the period $[k, k+1[$ and we have an arc (j, k) if and only if $r_j \leq k \leq d_j - 1$. Clearly, a feasible solution of $P \mid p_j = 1; r_j; d_j; pmtn \mid -$ provides a feasible flow. On the other hand, a feasible flow x_{jk} provides a feasible schedule by processing job j in $[k, k+1[$ for x_{jk} time units. All x_{jk} for $j = 1, \dots, n$ can be scheduled in $[k, k+1[$ on m machines due to McNaughton's wrap-around-rule, because $x_{jk} \leq 1$ for $j = 1, \dots, n$ and $\sum_{j=1}^n x_{jk} \leq m$.

The Lemma now follows from the fact (cf. Chvatal [4], Theorem 21) that if the transshipment problem has a feasible solution it also has an integer feasible solution. $\square$

A direct consequence of this Lemma is the following Theorem.

Theorem 1 : For problem $P \mid p_j = 1; r_j \mid \sum w_j U_j$ preemptions provide no better solution.

Proof: Consider an optimal solution for $P \mid p_j = 1; r_j; pmtn \mid \sum w_j U_j$ and let S be the set of jobs which are scheduled early in this optimal solution. Then according to Lemma 1 the jobs in S can also be scheduled early in a nonpreemptive schedule. Thus, an optimal solution of $P \mid p_j = 1; r_j \mid \sum w_j U_j$ is also optimal for the corresponding preemptive problem. $\square$

A corresponding result can be shown when intree-precedences are given:

Lemma 2 : For problem $P \mid p_j = 1; intree \mid \sum w_j U_j$ preemptions are redundant.

Proof: Consider an optimal solution S of $P \mid p_j = 1; intree; pmtn \mid \sum w_j U_j$. Let L be the set of jobs scheduled early in S . The partial schedule S' of S consisting of

all jobs in L solves the preemptive time window problem for L with time windows $[0, d_i]$. Let $k = \max_{i \in L} d_i$. Then a reverse schedule of S' solves the $P \mid p_j = 1; r_j = k - d_j; \text{outtree}; \text{pmtn} \mid C_{\max} \leq k$ -problem Q for L . Similar to Brucker et al. [2] it can be shown that the problem $P \mid p_j = 1; r_j; \text{outtree}; \text{pmtn} \mid C_{\max} \leq k$ has an optimal nonpreemptive solution. Thus, Q has a nonpreemptive solution S'' and the reverse schedule of S'' is a nonpreemptive optimal solution of $P \mid p_j = 1; \text{intree}; \text{pmtn} \mid \sum w_j U_j$. $\square$

3 Problem $P \mid p_j = 1 \mid \sum T_j$

In this section we prove that preemption is redundant for problem $P \mid p_j = 1 \mid \sum T_j$. The prove is based on several properties of optimal preemptive and nonpreemptive schedules which are given in the following three lemmata. To simplify notation, we assume that the jobs are enumerated such that $d_1 \leq d_2 \leq \dots \leq d_n$.

Lemma 3 : For $P \mid p_j = 1; \text{pmtn} \mid \sum T_j$ an optimal schedule exists with $C_1 \leq C_2 \leq \dots \leq C_n$.

Proof: Let $d_i \leq d_j$ but $C_j < C_i$. If the jobs i and j are exchanged the T_i value will reduce by

$$\Delta_1 = \max\{0, \min\{C_i - C_j, C_i - d_i\}\}$$

and the T_j value will increase by

$$\Delta_2 = \max\{0, \min\{C_i - C_j, C_i - d_j\}\}.$$

Since $d_i \leq d_j$, we get $\Delta_2 \leq \Delta_1$ and, thus, the solution value does not increase. $\square$

In the following we will show that for problem $P \mid p_j = 1 \mid \sum T_j$ preemption does not help to improve the solution.

Let $\bar{S}$ be an optimal solution of the non-preemptive problem $P \mid p_j = 1 \mid \sum T_j$ and let $\bar{C}_1, \dots, \bar{C}_n$ be the corresponding completion times of the jobs. Since an optimal schedule for this problem is achieved by scheduling the jobs in order of non-decreasing due dates as early as possible, we may assume that for a job $i = km + l$ with $1 \leq l \leq m$, $k \geq 0$ we have:

$$\bar{C}_i = k + 1$$

and, thus,

$$i/m \leq \bar{C}_i = \lceil i/m \rceil. \quad (3.1)$$

Furthermore, for $i \geq m$ we get

$$\sum_{j=i-m+1}^i \bar{C}_j = l(k+1) + (m-l)k = i. \quad (3.2)$$

On the other hand, let $\tilde{S}$ be an arbitrary optimal solution of the preemptive problem $P \mid p_j = 1; pmpt \mid \sum T_j$ and let $\tilde{C}_1, \dots, \tilde{C}_n$ be the corresponding completion times of the jobs. Due to Lemma 3 we may assume $\tilde{C}_1 \leq \dots \leq \tilde{C}_n$. Furthermore we get:

Lemma 4 :

1. $\tilde{C}_i \geq \max\{1, i/m\}$
2. $\sum_{j=i-m+1}^i \tilde{C}_j \geq i$; for $i \geq m$

Proof:

1. This inequality follows directly from the available machine capacity.
2. In a first step we prove the inequality for $i = n$. Let L_j be the completion time on machine j for $j = 1, \dots, m$ in schedule $\tilde{S}$. We may assume that no idle times occur during the time interval $[0, L_j]$ on machine j and that $L_1 \geq \dots \geq L_m$. Since

$$\sum_{j=1}^m L_j = \sum_{j=1}^n p_j = n,$$

it is sufficient to prove that

$$\sum_{j=n-m+1}^n \tilde{C}_j \geq \sum_{j=1}^m L_j. \quad (3.3)$$

Since $L_1 \geq \dots \geq L_m$, up to time L_j the machines $1, \dots, j$ all are busy. Thus, just before time L_j at least j jobs are not completed and, therefore, the j -th largest completion time is at least L_j , i.e.

$$\tilde{C}_{n-j+1} \geq L_j.$$

This directly implies (3.3).

For the case $i < n$, we may consider the partial subschedule of $\tilde{S}$ consisting of only the jobs $1, \dots, i$. This subschedule can easily be transformed in a new schedule of the jobs $1, \dots, i$ without any idle times and without any increase of the completion times. To this new schedule the prove for the case $i = n$ can be applied. $\square$

In the following we show that the solution $\overline{S}$ is not worse than solution $\tilde{S}$. The basis of the proof is the following lemma, which shows that if for one specific job i the tardiness w.r.t. $\overline{S}$ is larger than the tardiness w.r.t. $\tilde{S}$, we can find an index $r < i$ such, that the sum of tardiness of the jobs $r, r+1, \dots, i$ w.r.t. $\overline{S}$ is smaller than or equal to the sum of tardiness of these jobs w.r.t. $\tilde{S}$.

Lemma 5 : For each job i we can find a job r , $i - m + 1 \leq r \leq i$ such that

$$\sum_{j=r}^i \overline{T}_j \leq \sum_{j=r}^i \tilde{T}_j$$

($\overline{T}_j$ denotes the tardiness w.r.t. solution $\overline{S}$ and $\tilde{T}_j$ the tardiness w.r.t. solution $\tilde{S}$).

Proof:

Case 1: $\overline{C}_i \leq \tilde{C}_i$

By choosing $r = i$ we get the result of the lemma.

Case 2: $\overline{C}_i > \tilde{C}_i$ and i is not late in $\tilde{S}$

Due to (3.1) and Lemma 4 we get:

$$\overline{C}_i = \lceil i/m \rceil \leq \lceil \tilde{C}_i \rceil.$$

Since d_i is integral, this implies that i is also not late in $\overline{S}$ and, thus, by choosing $r = i$ we get the result of the lemma.

Case 3: $\overline{C}_i > \tilde{C}_i$ and i is late in $\tilde{S}$

To determine the index r in this case, we will look at the m jobs $t := i - m + 1, i - m + 2, \dots, i = km + l$. The partial schedule of these jobs within the schedule $\overline{S}$ is given in Figure 3. Since job i is late in $\tilde{S}$ we get

$$d_i < \tilde{C}_i < \overline{C}_i = k + 1$$

and, thus,

$$d_t \leq d_{t+1} \leq \dots \leq d_i \leq k \leq \overline{C}_t \leq \dots \leq \overline{C}_i,$$

which implies that all jobs $t, \dots, i$ are not early in $\overline{S}$. Thus,

$$\begin{aligned} \overline{T} &:= \sum_{j=t}^i \overline{T}_j &= \sum_{j=t}^i \max\{0, \overline{C}_j - d_j\} \\ &= \sum_{j=t}^i (\overline{C}_j - d_j) &= \sum_{j=t}^i \overline{C}_j - \sum_{j=t}^i d_j \\ &= i - \sum_{j=t}^i d_j \end{aligned} \tag{3.4}$$

Figure 3: Partial schedule of the jobs $t, \dots, i$ in $\bar{S}$

(the last equality follows from (3.2)).

Subcase 3a: $\tilde{C}_j \geq d_j$ for all $j = t, \dots, i$

In this case all jobs $t, \dots, i$ also are not early in $\tilde{S}$ and, thus, we get:

$$\begin{aligned} \tilde{T} := \sum_{j=t}^i \tilde{T}_j &= \sum_{j=t}^i \max\{0, \tilde{C}_j - d_j\} \\ &= \sum_{j=t}^i \tilde{C}_j - \sum_{j=t}^i d_j \geq i - \sum_{j=t}^i d_j \end{aligned} \tag{3.5}$$

(the last inequality follows from Lemma 4).

Combining (3.4) and (3.5) gives that the sum of the tardiness of the jobs $t, \dots, i$ for schedule $\bar{S}$ does not exceed that of schedule $\tilde{S}$. Thus, choosing $r = t = i - m + 1$ proves the lemma in this subcase.

Subcase 3b: There exists a job s in the set $\{t, \dots, i - 1\}$ for which $\tilde{C}_s < d_s$ holds, i.e. job s is early in $\tilde{S}$.

Let s be the job which largest index for which the above inequality holds. This implies that

$$\tilde{C}_j \geq d_j \text{ for } j = s + 1, \dots, i.$$

Furthermore, we have

$$\tilde{C}_j \leq \tilde{C}_s < d_s \leq k \leq \bar{C}_j \text{ for } j = t, \dots, s. \tag{3.6}$$

This yields:

$$\begin{aligned}
\sum_{j=s+1}^i \tilde{T}_j &= \sum_{j=s+1}^i \tilde{C}_j - \sum_{j=s+1}^i d_j \\
&= \sum_{j=t}^i \tilde{C}_j - \sum_{j=t}^s \tilde{C}_j - \sum_{j=s+1}^i d_j \\
&\geq i - \sum_{j=t}^s \tilde{C}_j - \sum_{j=s+1}^i d_j = \sum_{j=t}^i \bar{C}_j - \sum_{j=t}^s \tilde{C}_j - \sum_{j=s+1}^i d_j \\
&= \sum_{j=s+1}^i (\bar{C}_j - d_j) + \sum_{j=t}^s (\bar{C}_j - \tilde{C}_j) \\
&\geq \sum_{j=s+1}^i \bar{T}_j
\end{aligned}$$

(the last inequality follows from (3.6)). Thus, choosing $r = s + 1$ gives the stated result. $\square$

Using the above lemma, it is straightforward to show that for problem $P \mid p_j = 1 \mid \sum T_j$ preemption does not lead to better solutions.

Theorem 2 : For $P \mid p_j = 1 \mid \sum T_j$ preemptions provide no better schedule.

Proof: Assume that a preemptive schedule exists which is better than the optimal non-preemptive schedule $\bar{S}$. Let $\tilde{S}$ be such a better preemptive schedule for which the value i , for which

$$\sum_{j=i+1}^n \tilde{T}_j \geq \sum_{j=i+1}^n \bar{T}_j$$

holds, is minimal. Using Lemma 5 we get an index $r \leq i$ with $\sum_{j=r}^i \tilde{T}_j \geq \sum_{j=r}^i \bar{T}_j$. This implies

$$\sum_{j=r}^n \tilde{T}_j \geq \sum_{j=r}^n \bar{T}_j.$$

This is a contradiction to the minimality of i and, thus, the assumption that a better preemptive schedule exists does not hold. $\square$

4 Problem $P2 \mid p_j = 1; r_j \mid \sum w_j T_j$

In this section, we will show that for the case of two machines preemptions are redundant even if release dates are given and a more general objective function,

namely the total weighted tardiness, is considered. At first, this result is shown for the special case of instances with three jobs only.

Lemma 6 : For each instance of problem $P2 \mid p_j = 1; pmtn \mid \sum w_j T_j$ with three jobs a nonpreemptive schedule exists which is optimal.

Proof: Assume that the jobs are enumerated such that $d_1 \leq d_2 \leq d_3$ holds. Consider the following cases:

Case 1: $d_3 = 0$

In this case, it is $\sum w_j T_j = \sum w_j C_j$ and the result follows by a theorem of McNaughton [8].

Case 2: $d_3 \geq 2$

The schedule in which jobs 1 and 2 are processed in $[0, 1]$ and job 3 is processed in $[1, 2]$ is optimal. This follows from the fact that $T_3 = 0$ and that jobs 1,2 are scheduled optimally even for the preemptive problem.

Case 3: $d_3 = 1$

Assume that k is the job in $\{1, 2, 3\}$ with minimal weight. Then a schedule in which k is processed in $[1, 2]$ and the other two jobs in $[0, 1]$ is optimal. This follows from the fact that if job k would finish earlier, then $w_k T_k$ would decrease and the contribution of the other two jobs to the objective function would increase by at least the same amount (because $w_k = \min\{w_1, w_2, w_3\}$ and $d_1, d_2, d_3 \leq 1$ holds).

□

In the following, we use Lemma 6 to show that in the first time slot of an optimal schedule of problem $P2 \mid p_j = 1; r_j; pmtn \mid \sum w_j T_j$ there is no preemption.

Lemma 7 : Consider an instance of $P2 \mid p_j = 1; r_j; pmtn \mid \sum w_j T_j$ and let $r := \min_{j=1}^n r_j$. Assume that at least three jobs i with $r_i = r$ exist and at least four jobs with $r_i \leq r + 1$ exist. Then an optimal schedule exists in which two jobs are (non-preemptively) processed in $[r, r + 1[$.

Proof: Let S be an optimal preemptive schedule and set $I = \{i \mid r_i \leq r + 1\}$. Enumerate the jobs such that 1,2,3,4 are the jobs in I which finish first, second, third, and fourth in S , i.e. we have $C_1 \leq C_2 \leq C_3 \leq C_4$. Note, that $C_4 \geq r + 2$. Furthermore, $C_4 \leq C_i$ for all other jobs $i \in I$.

Let I' be the set of jobs which are partially scheduled in $[r, r + 2]$. As all release times are integers, we have $r_i \leq r + 1$ for all $i \in I'$. Thus, I' is a subset of I .

Now, we construct a new schedule by applying the following procedure.

1. Eliminate all pieces of jobs $i \in I' \cup \{1, 2, 3\}$ scheduled in $[r, C_4[$ and join pieces belonging to the same jobs to one partial job.
2. Schedule jobs 1,2,3 optimally in $[r, r + 2]$, two of them (say j and l) in $[r, r + 1]$ and the third (say k) in $[r + 1, r + 2]$ on M_1 .

Figure 4: Situation of the schedule after step 2

3. Sort the partial jobs from I' according to nonincreasing processing times and schedule them from left to right in the empty time slots in $[r + 1, C_4[$, first on M_1 and then on M_2 (apply McNaughton's wrapping around rule).

The resulting schedule is feasible since Job k protects a job which does not finish in the last empty time slot on M_1 from overlapping if this job is continued on M_2 at time $r + 1$. The new schedule is still optimal, because

- due to Lemma 6 jobs 1,2,3 are scheduled optimally even if preemption is allowed, and
- the finishing time of all other jobs does not increase (as $C_4 \leq C_i$ for all $i \in I' \subset I$).

□

Now, we can proceed by induction to prove:

Theorem 3 : For $P2 \mid p_j = 1, r_j \mid \sum w_j T_j$ preemptions provide no better schedule.

Proof: It is sufficient to show that an optimal schedule S for $P2 \mid p_j = 1; r_j; pmtn \mid \sum w_j T_j$ exists in which jobs are not preempted in $[r, r + 1[$. Then we conclude by induction that for the problem, which we get by eliminating the jobs scheduled in $[r, r + 1[$ in S , an optimal schedule without preemptions exist. This claim is clearly true for problems with $r_i = r$ for less than three jobs i or with less than four jobs i with $r_i \leq r + 1$. Otherwise the assertion follows by Lemma 7. $\square$

5 Concluding Remarks

We have shown that for the problems $P \mid p_j = 1; r_j \mid \sum w_j U_j$, $P2 \mid p_j = 1; r_j \mid \sum w_j T_j$, $P \mid p_j = 1 \mid \sum T_j$, and $P \mid p_j = 1;intree \mid \sum w_j U_j$ preemption is redundant. This settles the complexity status of the preemptive versions of these problems. The first three preemptive problems are polynomially solvable since their nonpreemptive counterparts reduce to network flow problems. The last preemptive problem is $\mathcal{NP}$ -hard because $1 \mid p_j = 1;chains \mid \sum U_j$ is already $\mathcal{NP}$ -hard (see Lenstra & Rinnooy Kan [7]). The question whether for $P \mid p_j = 1; r_j \mid \sum T_j$ or $P \mid p_j = 1 \mid \sum w_j T_j$ preemption is redundant as well is open.

It is important to note that for the new redundancy results with unit processing times the integrality assumption for the due dates is crucial. For rational due dates or, equivalently integer due dates and constant processing times ($p_j = p$) preemption is no longer redundant as the example of Fig. 2 shows. Even more surprisingly allowing preemption may turn a polynomially solvable problem into an $\mathcal{NP}$ -hard one. Brucker & Kravchenko [3] have shown that problem $P \mid p_j = p; pmtn \mid \sum w_j U_j$ is $\mathcal{NP}$ -hard although the corresponding nonpreemptive version can be solved in $O(n \log n)$. Sitters [9] proved a corresponding result for $R \mid pmtn \mid \sum C_j$. Finally, it is interesting to mention that preemption is nonredundant for $Q2 \mid p_j = 1 \mid \sum C_j$ (Baptiste & Timkovsky [1]). Together with the previously mentioned result for $P2 \mid p_j = 1 \mid C_{\max}$ preemptions may provide better solutions for all uniform machine problems.

References

- [1] **Baptiste, P. and Timkovsky, V. [2001]** On preemption redundancy in scheduling unit processing time jobs on two parallel machines, Operations Research Letters 28, 213–219.
- [2] **Brucker, P., Hurink, J. and Knust, S. [2001]** A polynomial algorithm for $P \mid p_j = 1, r_j, outtree \mid \sum C_j$, Osnabrücker Schriften zur Mathematik, Reihe P, No. 227.
- [3] **Brucker, P. and Kravchenko, S.A. [1999]** Preemption can make parallel machine scheduling problems hard, Osnabrücker Schriften zur Mathematik, Reihe P, No. 211.

- [4] **Chvatal, V. [1983]** Linear Programming, Freeman, New York.
- [5] **Du, J., Leung, J.Y.-T. and Young, G.H. [1991]** Scheduling chain-structured tasks to minimize makespan and mean flow time, *Information and Computation* 92, 219–236.
- [6] **Graham, R.L., Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G. [1979]** Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics* 5, 287–326.
- [7] **Lenstra, J.K. and Rinnooy Kan, A.H.G. [1980]** Complexity results for scheduling chains on a single machine, *European Journal of Operational Research* 4, 270–275.
- [8] **McNaughton, R. [1959]** Scheduling with deadlines and loss functions, *Management Science* 6, 1–12.
- [9] **Sitters, R.A. [2001]** Two NP-hardness results for preemptive minsum scheduling of unrelated parallel machines, *Proc. 8th International IPCO Conference, Lecture Notes in Computer Science*, Vol. 2081, Springer, 396–405.