Faculty of Mathematical Sciences

University of Twente

University for Technical and Social Sciences

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

Memorandum No. 1566

A polynomial algorithm for $P|p_j = 1, r_j, outtree|\Sigma C_j$ and $P|p_j = 1, r_j, outtree, pmtn|\Sigma C_j$

P. Brucker,[1] J.L. Hurink and S. Knust[1]

January 2001

ISSN 0169-2690

[1]Universität Osnabrück, Fachbereich Mathematik/Informatik, 49069 Osnabrück, Germany

A Polynomial Algorithm for $P \mid p_j = 1, r_j, outtree \mid \sum C_j$ and $P \mid p_j = 1, r_j, outtree, pmtn \mid \sum C_j$

Peter Brucker[1]
Universität Osnabrück, Fachbereich Mathematik/Informatik, 49069 Osnabrück, Germany
peter@mathematik.uni-osnabrueck.de

Johann Hurink
University of Twente, Faculty of Mathematical Sciences, P.O. Box 217, 7500 AE Enschede, The Netherlands
J.L.Hurink@math.utwente.nl

Sigrid Knust[1]
Universität Osnabrück, Fachbereich Mathematik/Informatik, 49069 Osnabrück, Germany
sigrid@mathematik.uni-osnabrueck.de

January 2001

Abstract: A polynomial algorithm is proposed for two scheduling problems for which the complexity status was open. A set of jobs with unit processing times, release dates and outtree precedence relations has to be processed on parallel identical machines such that the total completion time $\sum C_j$ is minimized. It is shown that the problem can be solved in $O(n^2)$ time if no preemption is allowed. Furthermore, it is proved that allowing preemption does not reduce the optimal objective value, which verifies a conjecture of Baptiste and Timkovsky.

Key words: scheduling, parallel machines, outtree, complexity results

Subject Classification: 90B35

[1]Supported by the DFG, Project 'Komplexe Maschinen-Schedulingprobleme'

1 Introduction

In this short note we consider two scheduling problems with parallel identical machines for which the complexity status was open. A set of n jobs $j = 1, \ldots, n$ with unit processing times $p_j = 1$ has to be processed on m parallel identical machines $M_1, \ldots, M_m$. Each machine can process at most one job at a time and each job can be processed on any of the machines. Integer release dates are given for the jobs which mean that job j cannot start before time r_j. Furthermore, precedence relations $i \rightarrow j$ among the jobs may be given which mean that job j cannot start before job i is completed. We restrict our considerations to outtree (outforest) precedences, i.e. each job has at most one predecessor. The problem is to find a feasible schedule (i.e. completion times C_j for all jobs j) minimizing the total completion time $\sum C_j$. We study the non-preemptive problem $P|p_j = 1, r_j, outtree| \sum C_j$ and its preemptive version $P|p_j = 1, r_j, outtree, pmtn| \sum C_j$. While in the first version no preemption of the jobs is allowed, in the second version the processing of the jobs may be interrupted at any time and resumed later on the same or another machine (but a job may not be processed on two different machines simultaneously).

Several complexity results exist for preemptive and non-preemptive parallel machine problems (cf. [2]). If no release dates are given, problem $P|p_j = 1, outtree| \sum C_j$ can be solved by Hu's level-algorithm for problem $P|p_j = 1, tree|C_{\max}$ (cf. Hu [4]). An interesting question concerning preemptive and non-preemptive problems is whether allowing preemption can reduce the optimal objective value. McNaughton [5] proved that preemptions are redundant in the case of arbitrary processing times, no precedences, no release dates and the weighted sum of completion times ($P|pmtn| \sum w_j C_j$). This result was strengthened by Du et al. [3] who showed that the result remains true if chain precedences are given ($P|chains, pmtn| \sum w_j C_j$). Recently, Baptiste & Timkovsky [1] showed that preemptions are not necessary for $P2|p_j = 1, r_j, outtree, pmtn| \sum C_j$ using rather complex interchange arguments. Furthermore, they conjectured that this result also holds for $m > 2$ machines. In this note we will prove their conjecture by, first, giving a polynomial algorithm for the open problem $P|p_j = 1, r_j, outtree| \sum C_j$ and, afterwards, showing that the resulting solution is also optimal for the preemptive version $P|p_j = 1, r_j, outtree, pmtn| \sum C_j$. The latter proof is much easier than the proof given by Baptiste & Timkovsky [1] for the two-machine case.

2 Problem $P \mid p_j = 1, r_j, outtree \mid \sum C_j$

In this section we present a polynomial algorithm for problem $P|p_j = 1, r_j, outtree| \sum C_j$. We assume that the release dates r_j are integer and compatible with the outtree precedence constraints, i.e. $r_i + 1 \leq r_j$ for all precedences $i \rightarrow j$.

In the following we will consider two relaxations of problem $P|p_j = 1, r_j, outtree| \sum C_j$. In the first we relax all precedence constraints, i.e. we consider problem $P|p_j =$

$1, r_j | \sum C_j$. Let $\tilde{r}_1 < \tilde{r}_2 < \ldots < \tilde{r}_k$ be the different release dates and define $\tilde{r}_{k+1} := \infty$. Denote by S_ν the set of jobs with release date $\tilde{r}_\nu$. An optimal schedule for this problem can be constructed by the following algorithm.

Algorithm $P|p_j = 1, r_j| \sum C_j$

```
1.  S := ∅;
2.  FOR ν := 1 TO k DO BEGIN
3.      t := r̃_ν;   S := S ∪ S_ν;
4.      WHILE t < r̃_{ν+1} and S ≠ ∅ DO BEGIN
5.          m_t := min {m, |S|};
6.          Schedule a set S_t ⊆ S of m_t jobs at time t;
7.          S := S \ S_t;  t := t + 1;
        END
    END
```

In the for-loop of this algorithm jobs are scheduled in the time periods $[\tilde{r}_\nu, \tilde{r}_{\nu+1}]$. This is done by first adding the set S_ν to the set of jobs which are available at time $t = \tilde{r}_\nu$ and then scheduling a maximal number of jobs from S in each of the next time periods. For all time periods t where m_t is not defined by the algorithm let $m_t := 0$.

In the second relaxation we replace the number m of machines by the number of jobs n. In this relaxation we can schedule all jobs from the set S_ν at time $\tilde{r}_\nu$ ($\nu = 1, \ldots, k$). The resulting schedule is feasible (since the release dates are compatible with the precedences) and certainly optimal for the problem with n machines. Let

$$\hat{m}_t := \begin{cases} |S_\nu|, & \text{if } t = \tilde{r}_\nu \\ 0, & \text{otherwise.} \end{cases}$$

To construct an optimal schedule for the original problem we transform the schedule of the second relaxation in which $\hat{m}_t$ jobs are scheduled in each time period $[t, t+1]$ into a feasible schedule where exactly m_t jobs are scheduled in period t (without violating the precedence constraints). The resulting schedule must be optimal for the original problem since it has the same objective value as the optimal schedule for the first relaxation.

This transformation is done by iteratively moving jobs from left to right. During the transformation the following invariance properties are satisfied:

(i) The outtree precedences are respected,

(ii) $\sum_{\nu \le \tau} \hat{m}_\nu \ge \sum_{\nu \le \tau} m_\nu$ holds for all time indices τ.

Clearly, these invariance properties hold at the beginning.

Assume that $\hat{m}_\nu < m_\nu$ for some index ν holds. Otherwise, $n = \sum_\nu \hat{m}_\nu \geq \sum_\nu m_\nu = n$ implies $\hat{m}_\nu = m_\nu$ for all ν and we are finished. Let t be the smallest time index with $\hat{m}_t < m_t \leq m$. Due to (ii) we must have an index $t' < t$ with $\hat{m}_{t'} > m_{t'}$ and $\hat{m}_\tau = m_\tau$ for $t' < \tau < t$. Furthermore, due to the definition of m_t we have $m_\tau = m$ for $t' < \tau < t$ which implies

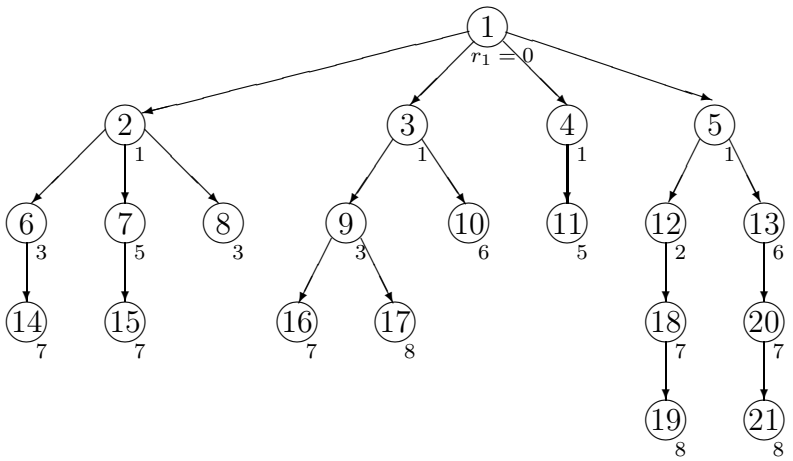
$$\hat{m}_{t'} \geq \hat{m}_{t'+1} = \hat{m}_{t'+2} = \ldots = \hat{m}_{t-1} > \hat{m}_t.$$

We successively will move a job from time period $[\tau - 1, \tau]$ to $[\tau, \tau + 1]$ for $\tau = t, t - 1, \ldots, t' + 1$ such that (i) keeps satisfied. This can be established as follows.

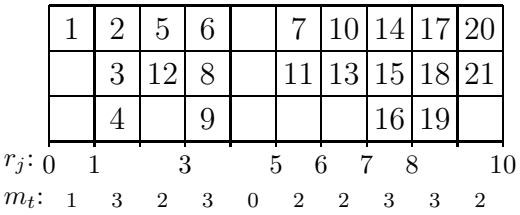
Since in $[\tau - 1, \tau]$ at least one more job is scheduled than in $[\tau, \tau + 1]$ and each job has at most one predecessor, we can always find in the interval $[\tau - 1, \tau]$ a job without successors in $[\tau, \tau + 1]$ which can be moved one time unit to the right. Also (ii) will be satisfied for t' because $\sum_{\nu<t'} \hat{m}_\nu \geq \sum_{\nu<t} m_\nu$ and $\hat{m}_\nu > m_\nu$. Thus, after the move of one job from $[t', t' + 1]$ to $[t' + 1, t' + 2]$ the inequality (ii) still holds for $\tau = t'$. It also holds for all $\tau > t'$ because by the transformation the $\hat{m}_\nu$-values do not decrease for $\nu > t'$.

In one step of the transformation the value of $\max\{0, m_\nu - \hat{m}_\nu\}$ is decreased by one for $\nu = t$ and kept unchanged for all other ν. Thus, $\sum_\nu \max\{0, m_\nu - \hat{m}_\nu\}$ is decreased by one in each step. Due to the fact that at the beginning $\sum_\nu \max\{0, m_\nu - \hat{m}_\nu\} \leq \sum_\nu m_\nu = n$, after at most n iterations we reach an optimal schedule for the original problem. Since each iteration can be implemented in $O(n)$ time, the overall complexity to solve problem $P|p_j = 1, r_j, outtree|\sum C_j$ is $O(n^2)$.

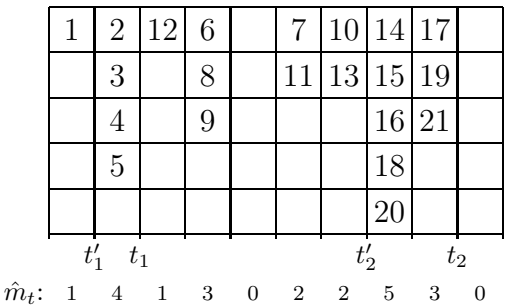
Example: Consider an instance of $P|p_j = 1, r_j, outtree|\sum C_j$ with $m = 3$ machines, $n = 21$ jobs and the following outtree precedences and release dates:



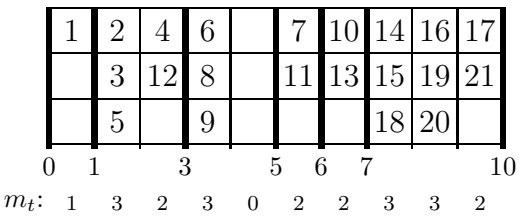
Applying `Algorithm` $P|p_j = 1, r_j|\sum C_j$ to this instance produces the schedule shown in Figure 1(a) in which the precedences $5 \to 12, 18 \to 19$ and $20 \to 21$ are violated. An optimal schedule for $m = n$ machines is presented in Figure 1(b) and an optimal transformed schedule can be found in Figure 1(c). In this transformation job 4 is moved from the interval $[1, 2]$ to the interval $[2, 3]$, jobs 17, 21 are moved from $[8, 9]$ to $[9, 10]$ and jobs 16, 20 are moved from $[7, 8]$ to $[8, 9]$ (cf. Figure 1(b) and (c)).



(a) Infeasible schedule obtained by `Algorithm` $P|p_j = 1, r_j|\sum C_j$



(b) Optimal schedule for $m = n$ machines



(c) Optimal feasible schedule for $m = 3$ machines

Figure 1: Schedules for problem $P|p_j = 1, r_j, outtree|\sum C_j$

3 Problem $P \mid p_j = 1, r_j, outtree, pmtn \mid \sum C_j$

In this section we show that for problem $P|p_j = 1, r_j, outtree, pmtn|\sum C_j$ an optimal schedule exists in which preemption is not necessary. Thus, the algorithm derived in the previous section also solves problem $P|p_j = 1, r_j, outtree, pmtn|\sum C_j$.

Our claim is a consequence of the following facts:

(1) The optimal solution value for the relaxation $P|p_j = 1, r_j, pmtn|\sum C_j$ is a lower bound for the corresponding problem $P|p_j = 1, r_j, outtree, pmtn|\sum C_j$.

(2) Corresponding instances of $P|p_j = 1, r_j, pmtn|\sum C_j$ and $P|p_j = 1, r_j|\sum C_j$ have the same optimal solution value.

(3) Given an instance I of $P|p_j = 1, r_j, outtree|\sum C_j$ where the release dates are compatible with the outtree-precedences, an optimal solution of the related instance of $P|p_j = 1, r_j|\sum C_j$ with the same release dates can be transformed into a feasible solution of I without changing the objective value.

Clearly, (1) holds. (3) has been shown in the previous section. To prove (2) we decompose the schedule constructed by `Algorithm` $P|p_j = 1, r_j|\sum C_j$ (or the corresponding schedule for $P|p_j = 1, r_j, outtree|\sum C_j$) into **blocks**. These blocks are defined as follows.

Let $i_0 = 1 < i_1 < \ldots < i_q$ be indices such that for $\nu = 0, \ldots, q-1$

- $i_{\nu+1}$ is the first index greater than i_ν such that `Algorithm` $P|p_j = 1, r_j|\sum C_j$ schedules all jobs j with $\tilde{r}_{i_\nu} \leq r_j < \tilde{r}_{i_{\nu+1}}$ within the interval $[\tilde{r}_{i_\nu}, \tilde{r}_{i_{\nu+1}}[$, and
- at least one machine is idle for at least one time unit in $[\tilde{r}_{i_\nu}, \tilde{r}_{i_{\nu+1}}[$.

Note that jobs of block ν cannot be scheduled before the corresponding interval $[\tilde{r}_{i_\nu}, \tilde{r}_{i_{\nu+1}}[$ since their release dates are greater or equal to $\tilde{r}_{i_\nu}$. The partial schedule corresponding to the interval $[\tilde{r}_{i_\nu}, \tilde{r}_{i_{\nu+1}}[$ is called a block. In the example in Figure 1(c) we have six blocks in the intervals $[0, 1[, [1, 3[, [3, 5[, [5, 6[, [6, 7[$ and $[7, 10[$.

Now (2) can be proved by induction on the number of blocks. If only one block exists, then consider the relaxed problem $P|p_j = 1, pmtn|\sum C_j$ in which all release dates are ignored. McNaughton [5] showed that for this problem without release dates preemption is not necessary, i.e. for the block an optimal non-preemptive schedule exists. The resulting schedule fits completely in the time interval corresponding to the block. Furthermore, an optimal solution of $P|p_j = 1|\sum C_j$ has the same structure as the solution provided by `Algorithm` $P|p_j = 1, r_j|\sum C_j$ which is feasible for $P|p_j = 1, r_j, pmtn|\sum C_j$. Thus, the latter solution must be optimal for $P|p_j = 1, r_j, pmtn|\sum C_j$.

Assume that (2) holds for an instance with $k-1$ blocks and consider an instance with k blocks. Consider the relaxation of this problem in which the first block and the remaining $k-1$ blocks are optimized separately. By induction preemption is redundant for each of these subproblems and the corresponding schedules fit into their time intervals. Thus, both schedules do not overlap, i.e. the composed schedule is feasible for the problem with k blocks. Consequently, the composed schedule is an optimal non-preemptive schedule for the preemptive problem.

4 Concluding remarks

We have presented an $O(n^2)$-algorithm for problem $P|p_j = 1, r_j, outtree|\sum C_j$ and shown that preemptions in $P|p_j = 1, r_j, outtree, pmtn|\sum C_j$ are redundant. Thus, the algorithm also solves the preemptive problem. However, this result does not hold for other related problems. Baptiste & Timkovsky [1] showed that preemptions become advantageous for the following problems: $P2|p_j = 1, pmtn|C_{\max}$, $P2|p_j = 1, intree, pmtn|\sum C_j$, $P2|p_j = 1, outtree, pmtn|\sum w_j C_j$, and $Q2|p_j = 1, pmtn|\sum C_j$.

References

[1] **Baptiste, P. and Timkovsky, V. [2000]** On preemption redundancy in scheduling unit processing time jobs on two parallel machines, Technical Report 2000-348, Universite de Technologie de Compiegne, France.

[2] **Brucker, P. and Knust, S.** Complexity results for scheduling problems, `http://www.mathematik.uni-osnabrueck.de/research/OR/class/.`

[3] **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.

[4] **Hu, T.C. [1961]** Parallel sequencing and assembly line problems, Operations Research 9, 841–848.

[5] **McNaughton, R. [1959]** Scheduling with deadlines and loss functions, Management Science 6, 1–12.