Math Meth Oper Res (1999) 49:211–219

Mathematical Methods
of Operations Research

Scheduling identical jobs with chain precedence constraints on two uniform machines

Peter Brucker[1,*], Johann Hurink[2], Wieslaw Kubiak[3,]**

[1] Universität Osnabrück, Fachbereich Mathematik/Informatik, D-49069 Osnabrück, Germany (e-mail: peter@mathematik.uni-osnabrueck.de)
[2] University of Twente, Faculty of Mathematical Sciences, P.O. Box 217, 7500 AE Enschede, The Netherlands (e-mail: j.l.hurink@math.utwente.nl)
[3] Memorial University of Newfoundland, Faculty of Business Administration, St. John's, Newfoundland, Canada (e-mail: wkubiak@morgan.ucs.mun.ca)

Abstract. The problem of scheduling identical jobs with chain precedence constraints on two uniform machines is considered. It is shown that the corresponding makespan problem can be solved in linear time.

Key words: Scheduling, uniform machines, identical jobs, chain precedence constraints

1 Introduction

We consider the problem $Q|prec,\ p_j = 1|C_{\max}$ of scheduling identical jobs with precedence constraints on m uniform machines $M_1, \dots, M_m$ with the objective to minimize the makespan. Each job has the same processing time p_i on machine M_i. If there are no precedence constraints between the jobs this problem can be solved in $O(n \log m)$ time (Lawler et al. [1993]) even for the objective functions $\sum_{i=1}^{n} f_i(C_i)$ and $\max_{i \in \{1,\dots,n\}} f_i(C_i)$ where f_i is a monotone function of the finish time C_i of job i. Despite the fact that there is an $O(n^6)$-algorithm for problem $Q2|pmtn,\ prec,\ r_j|L_{\max}$, where jobs with arbitrary processing times, release times, and arbitrary precedence constraints are to be processed preemptively on two uniform machines to minimize maximum lateness (Lawler [1982]), only two polynomial algorithms have been developed for special precedence constraints. Namely, Kubiak [1989] gives a polynomial time algorithm for problem $Q2|tree,\ p_j = 1|C_{\max}$ with one processor b times

* Supported by Deutsche Forschungsgemeinschaft, Project Br 389/15-1, 'Komplexe Machinen-Schedulingprobleme'.
** Supported by the Natural Sciences and Engineering Research Council of Canada under Grant OGP0105675, and by the Komitet Badan Naukowych of Poland under Grant 8T11C04012.
Manuscript received: February 1997/final version received: May 1998

faster than the other one, where b is integer. Gabow [1982] tackles the same problem for $b = 1 + 1/k$, where k is an integer. The complexity of the corresponding problem with arbitrary rational b is unknown.

In this paper we will present an algorithm for problem $Q2|chains, p_j = 1|C_{\max}$ with two uniform processors, identical jobs, chain precedence constraints, and makespan minimization. The algorithm works in $O(k)$ time, where k is the number of chains. The results give some insight in the loss that may be incurred in case of the problem $Q2|pmtn|C_{\max}$ (with integer processing times) when preemption is allowed at integral points of time only.

Throughout this paper we assume that M_2 is faster than M_1 and that $p_2 = p < p_1 = 1$ where p is a rational number. Furthermore, we assume that we have k chains with $n_1 \le n_2 \le \cdots \le n_k$ jobs in each chain, where $n = \sum_{j=1}^{n} n_j$ is the total number of jobs.

2 The solution procedure

In Sections 2.1 and 2.2 we will discuss solutions to two relaxations of $Q2|chains, p_j = 1|C_{\max}$, namely, $Q2|chains, pmtn, p_j = 1|C_{\max}$ (which is equivalent to $Q2|pmtn|C_{\max}$ with integer processing times) and $Q2|p_j = 1|C_{\max}$. These solutions are useful guidelines for the solution procedure for $Q2|chains, p_j = 1|C_{\max}$ presented in Section 2.3.

2.1 k chains with preemption

To solve problem $Q2|chains, pmtn, p_j = 1|C_{\max}$ three cases are considered. These cases and the corresponding optimal solutions are shown in Figure 1. Notice, that at most two preemptions may be necessary in an optimal schedule for $Q2|chains, pmtn, p_j = 1|C_{\max}$. Therefore, if we delete the preempted jobs from machine M_1 and add them at the end of the schedule on M_2, then we obtain a schedule which is at most $2p$ away from the optimum for $Q2|chains, p_j = 1|C_{\max}$. However, a better solution for $Q2|chains,$

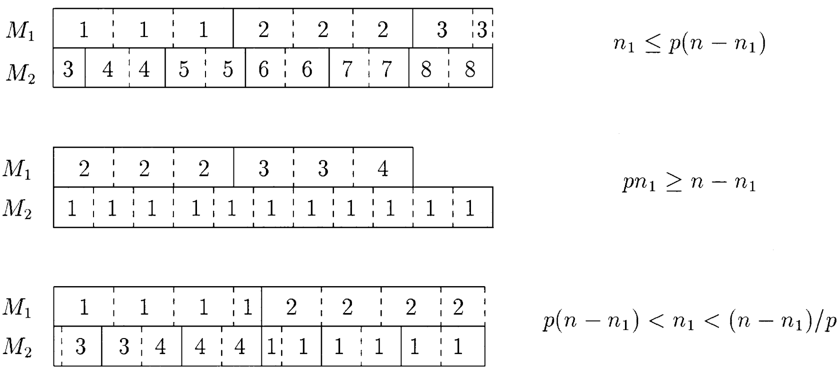

Fig. 1. Three cases for $Q2|chains, pmtn, p_j = 1|C_{\max}$

$p_j = 1|C_{\max}$ may be obtained. To derive an optimal solution the second relaxation is quite useful.

2.2 Independent jobs

Now we consider the problem $Q2|p_j = 1|C_{\max}$ with independent jobs. If we schedule y jobs on M_2 (and thus $n - y$ jobs on M_1) then the makespan is given by

$$\max\{py, n - y\}. \tag{2.1}$$

We need to find an integer y, $0 \le y \le n$, which minimizes (2.1). This is accomplished by solving equation $py = n - y$ and rounding up or down its solution $\bar{y} = \dfrac{n}{p+1}$ depending on which (rounded) value minimizes (2.1). Thus, the optimal solution y^* of (2.1) is given by

$$y^* = \begin{cases} \left\lceil \dfrac{n}{p+1} \right\rceil & \text{if } n - \left\lfloor \dfrac{n}{p+1} \right\rfloor \ge p \left\lceil \dfrac{n}{p+1} \right\rceil \\ \left\lfloor \dfrac{n}{p+1} \right\rfloor & \text{otherwise} \end{cases}$$

(see Fig. 2). The corresponding makespan $\bar{C}$ is given by py^* if $y^* = \left\lceil \dfrac{n}{p+1} \right\rceil$ and by $n - \left\lfloor \dfrac{n}{p+1} \right\rfloor$ in the other case. Clearly y^* and $\bar{C}$ can be calculated in constant time.

2.3 k chains

Our idea to solve the k-chain problem is as follows. First, we will solve the relaxed problem, i.e. $Q2|p_j = 1|C_{\max}$, where we consider all jobs to be inde-

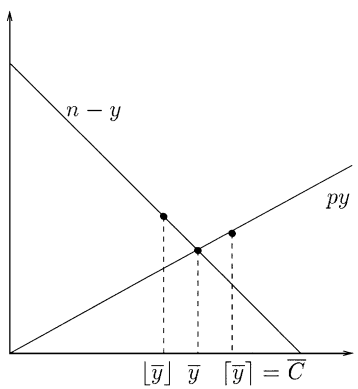

Fig. 2. Functions defining the makespan according to (2.1)

pendent. Based on the optimal solution of this relaxation we will give some feasible schedules for $Q2|chains,\ p_j = 1|C_{\max}$ for which we can show that all other schedules are dominated by at least one of them. Therefore, the shortest of the given schedules is optimal. In the following we will give a detailed description of the procedure.

Let k chains $1, \dots, k$ with $n_1 \geq n_2 \geq \cdots \geq n_k$ jobs be given and let $n = \sum_{j=1}^{k} n_j$. Furthermore, let $m_1 (m_2)$ be the number of jobs on $M_1 (M_2)$ in an optimal solution for the corresponding relaxed problem with n independent jobs. In the following we assume $pn_1 < \bar{C}$, since otherwise the solution, where chain 1 is scheduled on M_2 and the remaining chains are scheduled on M_1 has makespan pn_1 and, thus, is optimal.

Let $x_1 \leq n_1$ be the maximal integer with

$$x_1 + p(n_1 - x_1) \leq \bar{C},$$

i.e.

$$x_1 = \left\lfloor \frac{\bar{C} - pn_1}{1 - p} \right\rfloor. \tag{2.2}$$

If $x_1 = n_1$, it is possible to schedule all jobs of chain 1 on M_1. Therefore, we may schedule the chains in order $1, \dots, k$ in a wrap around manner. First, we schedule jobs on M_1 starting with the jobs of chain 1 and continuing with the jobs of chains $2, 3, \dots$ until m_1 jobs have been scheduled on M_1. Let the job scheduled last on M_1 belong to chain i. We continue by scheduling the remaining m_2 jobs on M_2 starting with the remaining jobs of chain i and continuing with the jobs of chains $i + 1, \dots, k$. If we reorder the jobs of chain i in such a way that they respect the precedence constraints, the resulting schedule is feasible and has makespan $\bar{C}$, thus, it must be optimal.

It remains to consider the case $0 \leq x_1 \leq n_1$ in more detail. In this case a feasible schedule of chain 1 is given in Figure 3. From (2.2) it follows that the gap Δ between the last job of chain 1 on M_1 and first job of chain 1 scheduled on M_2 is smaller that $1 - p < 1$.

If $n_k \leq m_1 - x_1$, we can extend the schedule of Figure 3 to a schedule with makespan $\bar{C}$ as follows. Schedule the jobs of chain k directly after the jobs of chain 1 on M_1 (see Figure 4(a)). Afterwards, the jobs of the remaining chains are scheduled arbitrarily in the remaining $m_1 - (x_1 + n_k)$ positions on M_1 and in the $m_2 - (n_1 - x_1)$ positions before the jobs of chain 1 on M_2. Since the first job of chain k on M_1 covers the gap between x_1 and $x_1 + \Delta$, the resulting schedule is feasible and has makespan $\bar{C}$.

If $n_k > m_1 - x_1$, we extend the schedule of Figure 3 by scheduling $x_k := m_1 - x_1$ jobs of chain k starting at time $\bar{C}$ on M_1 from right to left. Afterwards, we schedule the remaining $n_k - x_k$ jobs of chain k starting at time 0 on M_2. There are two possible outcomes:

Case 1: The jobs of chain k do not overlap (see Figure 4(b)).
In this case, the schedule of Figure 4(b) can be completed by scheduling the jobs of the remaining chains arbitrarily in the free positions between the jobs of chains k and 1 on M_2. Since the jobs of chain k do not overlap, the resulting schedule is feasible and has makespan $\bar{C}$.

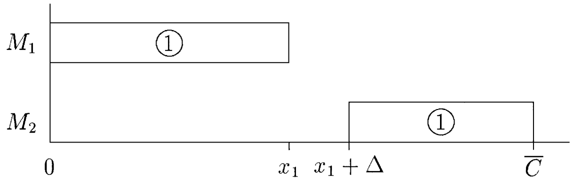

Fig. 3. Schedule of the jobs of chain 1

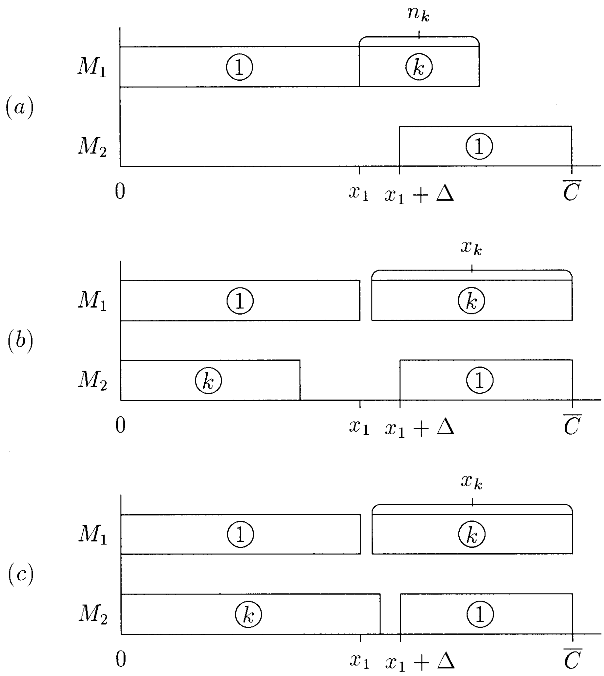

Fig. 4. Schedule of the job of chain 1 and k

Case 2: The jobs of chain k overlap (see Figure 4(c)).
Since the jobs of chain k overlap, we must have $x_k \geq 1$. In further considerations we distinguish two cases.

Case 2.1: $k = 2$
We will determine a solution departing from the infeasible schedule of Figure 4(c). In fact we will construct three feasible schedules for the jobs of chains 1 and 2 such that all other feasible schedules are dominated by at least one of them. The three schedules are given in Figure 5.

i) For schedule $S1$ in Figure 5 we have

$$C_{\max}(S_1) = x_2 + (n_2 - x_2)p. \tag{2.3}$$

We can conclude:

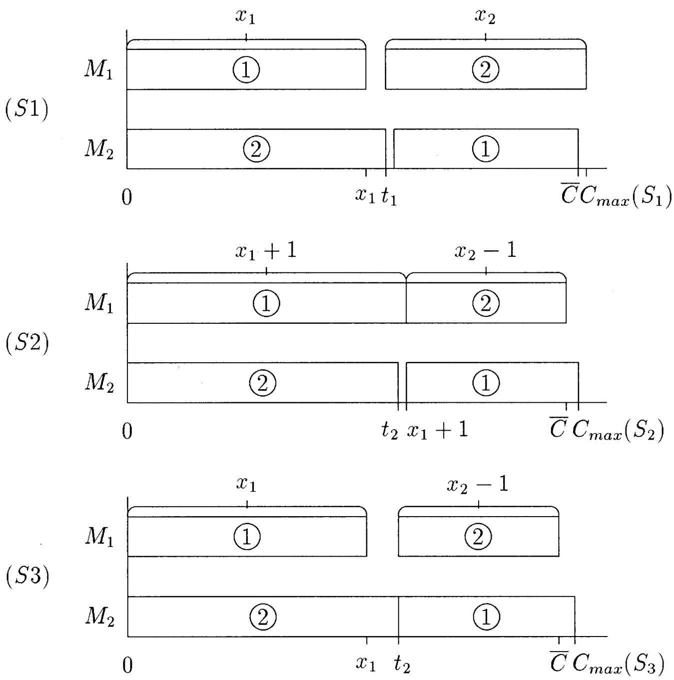

Fig. 5. Three feasible schedules for two chains

Dominance 1: Schedule $S1$ is at least as good as any feasible schedule with at least x_2 jobs of chain 2 on M_1.

ii) For schedule $S2$ in Figure 5 we have

$$C_{\max}(S_2) = x_1 + 1 + (n_1 - x_1 - 1)p. \tag{2.4}$$

Schedule $S2$ is feasible since

$$t_2 = t_1 + p \leq x_1 + \Delta + p < x_1 + 1 - p + p = x_1 + 1$$

and we can conclude:

Dominance 2: Schedule $S2$ is at least as good as any feasible schedule with at least $x_1 + 1$ jobs of chain 1 on M_1.

iii) For schedule $S3$ is Figure 5 we have

$$C_{\max}(S_3) = (m_2 + 1)p. \tag{2.5}$$

Schedule $S3$ is feasible since

$$x_2 - 1 \leq \bar{C} - (x_1 + 1) < \bar{C} - (x_1 + \Delta) = p(n_1 - x_1).$$

We can conclude:

Dominance 3: Schedule $S3$ is at least as good as any feasible schedule with at least $m_2 + 1$ jobs on M_2.

Lemma 1. *The best of the three schedules S_1, S_2 and $S3$ is optimal.*

Proof: Let now an arbitrary feasible schedule S of the two chains 1 and 2 be given and let $\tilde{x}_1(\tilde{x}_2)$ denote the number of jobs of chain 1(2) on M_1 in this schedule. If $\tilde{x}_1 \geq x_1 + 1$ schedule S is dominated by $S2$ (see Dominance 2) and if $\tilde{x}_2 \geq x_2$, schedule S is dominated by $S1$ (see Dominance 1). It remains to consider the case $\tilde{x}_1 \leq x_1$ and $\tilde{x}_2 \leq x_2 - 1$. However, in this case we have $\tilde{x}_1 + \tilde{x}_2 \leq x_1 + x_2 - 1 = m_1 - 1$ and, therefore, at least m_2+1 jobs are scheduled on M_2 in S. Thus, schedule S is dominated by $S3$ (see Dominance 3). Summarizing, we can state that each feasible schedule is dominated by at least one of the feasible schedules $S1, S2$, and $S3$. Thus, the best of the three schedules $S1, S2$ and $S3$ is an optimal schedule.

Case 2.2: $k \geq 3$
We will distinguish two cases based on the number x_1 of jobs of chain 1 on M_1 in the schedule given in Figure 4(c).

Case 2.2.1: $x_1 \geq 1$
In a feasible schedule for the independent job problem, the jobs of chains $2, \ldots, k-1$ would have to be inserted in the gap between the jobs of chain k and chain 1 on M_2. However, this gap is smaller or equal to $\Delta < 1$. Thus, with $n_2 p \geq n_k p \geq x_1 \geq 1$ we get a contradiction, and Case 2.2.1 can not occur.

Case 2.2.2: $x_1 = 0$
In the following we will consider two schedules $S4$ and $S5$. Schedule $S4$ with makespan $C_{\max}(S4) = np$ schedules all jobs of all chains on M_2 starting at time 0. Schedule $S5$ is constructed in the following way (see Figure 6):

- schedule one job of chain k starting at time 0 on M_1
- schedule the remaining jobs of chain k and the jobs of chain 1 as last jobs on M_2, where the jobs of chain 1 are scheduled before the jobs of chain k

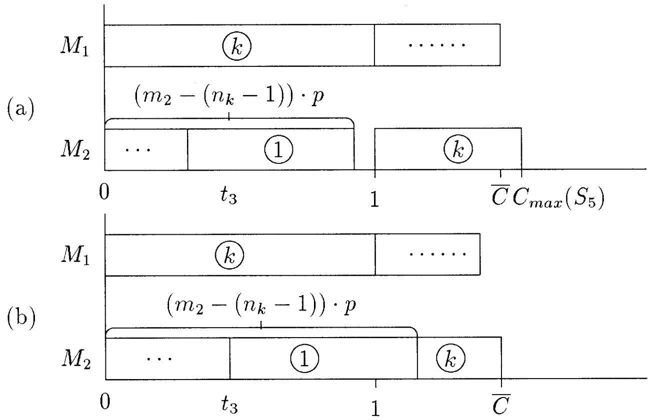

Fig. 6. Two possible schedules $S5$

- schedule the jobs of the remaining chains $2, \ldots, k-1$ arbitrarily in the remaining $m_1 - 1$ positions on M_1 and $m_2 - (n_1 + n_k - 1)$ position on M_2 before chain 1.

In this schedule jobs of the chains $2, \ldots, k-1$ will not overlap since the first job of these chains on M_1 starts at time 1 and the last job of these chains on M_2 finishes at time t_3 which is bounded as follows:

$$t_3 = p(m_2 - (n_1 + n_k - 1)) = m_2 p - (n_1 + n_k - 1)p$$
$$\leq \bar{C} - n_1 p - (n_k - 1)p \leq \bar{C} - n_1 p < 1.$$

Therefore, if we do not start the last job of chain k on M_2 before 1, then the schedule S5 will be feasible.

The two possible outcomes of schedule $S5$ are given in Figure 6. In case (a) of Figure 6, chain k determines the makespan, which is given by $C_{\max}(S_5) = 1 + (n_k - 1)p$. Since k is the shortest chain, each schedule with at least one job on M_1 must have a makespan greater or equal $C_{\max}(S_5)$. In case (b) of Figure 6, schedule $S5$ has makespan $\bar{C}$, and therefore $S5$ is optimal. Summarizing, we get

Dominance 4: Schedule $S5$ is at least as good as any schedule with at least one job on M_1.

Therefore, the better of the two schedules $S4$ and $S5$ is an optimal schedule.

Summarizing, we can solve the k-chain problem by

- determining the total number n of jobs, the length n_1 of the longest chain, and the length n_k of the shortest chain ($O(k)$ time),
- solving an independent job problem with n jobs (constant time), and
- carrying through the case analysis described in 2.2 (constant time).

Thus, the overall complexity is $O(k)$. However, with the knowledge of n, n_1, n_k the problem can be solved in constant time. Notice, that the presented approach works also if p is a real number.

3 Concluding remarks

We have presented a linear time algorithm for problem $Q2|chains, p_j = 1|C_{\max}$. The complexity of problem $Q2|tree, \ p_i = 1|C_{\max}$, which we get by replacing the chains by a tree, and problem $Q3|chains, \ p_j = 1|C_{\max}$, which we get by enlarging the number of machines by one, are still open.

References

Gabow HN (1982) An almost-linear algorithm for two-processor scheduling. Journal Assoc. Comput. Mach. 29:766–780

Kubiak W (1989) Optimal schedules of unit-time tasks on two uniform processors under tree-like precedence constraints. ZOR 33:423–437

Lawler EL (1982) Preemptive scheduling of precedence-constrained jobs on parallel machines. In: Dempster MAH, Lenstra JK, Rinnooy Kan AHG (Eds.) Deterministic and stochastic scheduling, Reidel, Dordrecht

Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (1993) Sequencing and scheduling: algorithms and complexity. In: Graves SC, Rinnooy Kan AHG, Zipkin P (Eds.) Handbook in operations research and management science, Volume 4: Logistics of production and inventory, North-Holland, pp. 445–522