Preemptive Scheduling of a Single Machine to Minimize Maximum Cost Subject to Release Dates and Precedence Constraints

K. R. BAKER
Dartmouth College, Hanover, New Hampshire

E. L. LAWLER
University of California, Berkeley, California

J. K. LENSTRA
Mathematisch Centrum, Amsterdam, The Netherlands

A. H. G. RINNOOY KAN
Erasmus University, Rotterdam, The Netherlands

(Received September 1980; accepted November 1981)

Suppose n jobs are to be processed on a single machine, subject to release dates and precedence constraints. The problem is to find a preemptive schedule which minimizes the maximum job completion cost. We present an $O(n^2)$ algorithm for this problem, generalizing previous results of E. L. Lawler.

WE CONSIDER the problem of finding a preemptive schedule for n jobs on a single machine which minimizes the maximum job completion cost subject to given release dates and precedence constraints.

Specifically, suppose that n *jobs* $1, \cdots, n$ are to be processed on a *single machine* which can execute at most one job at a time. Each job j becomes available at its *release date* r_j and requires a *processing time* p_j. Moreover, a *precedence relation* $\rightarrow$ between the jobs is given; if $j \rightarrow k$, it is required that job j be completed before job k can start. Unlimited *preemption* is allowed, i.e., the execution of any job may arbitrarily often be interrupted and resumed at a later time. Associated with each job j is a monotone nondecreasing *cost function* f_j; if job j has a *completion time* C_j, a cost $f_j(C_j)$ is incurred. We assume that each f_j can be evaluated in unit time for any value of the argument. The problem is to find a feasible schedule such that the *maximum cost* $f_{\max} = \max_j\{f_j(C_j)\}$ is minimized.

If all release dates are equal, there is no advantage to preemption, and the problem is solvable in $O(n^2)$ time (see Lawler [1973]). In Section 1, we recall this algorithm and give an alternative proof of its correctness.

Subject classification: 432 computational complexity, 584 single machine scheduling.

Operations Research
Vol. 31, No. 2, March–April 1983

0030-364X/83/3102-0381 $01.25

If the release dates may be arbitrary, the preemptive case is still solvable in $O(n^2)$ time. In Sections 2 and 3, we present an algorithm for this case by extending the approach of Section 1; Section 4 contains a numerical example. We note that the nonpreemptive case is NP-hard in the strong sense, even without precedence constraints (see Garey and Johnson [1977], and Lenstra et al. [1977]).

In the notation of Graham et al. [1979], $1|prec|f_{\max}$ is considered in Section 1, $1|pmtn, r_j|f_{\max}$ in Section 2 and $1|pmtn, prec, r_j|f_{\max}$ in Section 3; $1|r_j|f_{\max}$ is unary NP-hard.

1. EQUAL RELEASE DATES

We first assume that all release dates are equal to 0, in which case we need only consider schedules without preemption and without machine idle time.

Let $N = \{1, \cdots, n\}$ be the set of all jobs, let $N' \subseteq N$ be the set of jobs without successors, and for any $S \subseteq N$ define

$$p(S) = \textstyle\sum_{j \in S} p_j;$$
$$f_{\max}(S) = \text{value of } f_{\max} \text{ in a given schedule for the jobs in } S;$$
$$f^*_{\max}(S) = \text{value of } f_{\max} \text{ in an optimal schedule for the jobs in } S.$$

Clearly, $f^*_{\max}(N)$ satisfies the following two inequalities: $f^*_{\max}(N) \geq \min_{j \in N'}\{f_j(p(N))\}$; and $f^*_{\max}(N) \geq \max_{j \in N}\{f^*_{\max}(N - \{j\})\}$.

Let job $l \in N'$ be such that $f_l(p(N)) = \min_{j \in N'}\{f_j(p(N))\}$. Consider a schedule which is optimal subject to the condition that job l is processed *last.* For any such schedule, we have

$$f_{\max}(N) = \max\{f_l(p(N)), \quad f^*_{\max}(N - \{l\})\} \leq f^*_{\max}(N).$$

It follows that there exists an optimal schedule in which job l is in the last position.

By repeated application of this rule, one obtains an optimal schedule in $O(n^2)$ time. The above correctness proof differs from the one given by Lawler in that it does not rely on an interchange argument.

2. ARBITRARY RELEASE DATES, NO PRECEDENCE CONSTRAINTS

We now consider the case in which arbitrary release dates are specified. First, we assume that there are no precedence constraints.

Our first observation is that, since preemption is allowed, it is never advantageous to leave the machine idle when unscheduled jobs are available. Hence, the time at which all jobs will be completed can be determined in advance by scheduling the jobs in order of nondecreasing r_j. This schedule naturally decomposes into *blocks*, where a block $B \subseteq N$ is defined as a minimal set of jobs processed without idle time from

$r(B) = \min_{j \in B}\{r_j\}$ until $t(B) = r(B) + p(B)$, such that each job $j \notin B$ is either completed not later than $r(B)$ ($C_j \leq r(B)$) or not released before $t(B)$ ($r_j \geq t(B)$).

It is easily seen that, in minimizing $f_{\max}$, we can consider each block B separately. As in Section 2, $f^*_{\max}(B)$ satisfies the following two inequalities: $f^*_{\max}(B) \geq \min_{j \in B}\{f_j(t(B))\}$; and $f^*_{\max}(B) \geq \max_{j \in B}\{f^*_{\max}(B - \{j\})\}$.

Let job $l \in B$ be such that

$$f_l(t(B)) = \min_{j \in B}\{f_j(t(B))\}. \quad (1)$$

Consider a schedule for block B which is optimal subject to the condition that job l is processed *only if no other job is available*. It consists of two complementary parts:

(i) An optimal schedule for the set $B - \{l\}$, which decomposes into a number of subblocks $B_1, \cdots, B_b$ with respect to this job set;
(ii) A schedule for job l, which is given by $[r(B), t(B)] - \cup_{i=1}^{b}[r(B_i), t(B_i)]$; note that job l is completed at time $t(B)$.

For any such schedule, we have

$$f_{\max}(B) = \max\{f_l(t(B)), f^*_{\max}(B - \{l\})\} \leq f^*_{\max}(B).$$

It follows that there exists an optimal schedule in which job l is scheduled as prescribed above.

The problem can now be solved in the following way. First, order the jobs according to nondecreasing r_j in $O(n \log n)$ time. Next, determine the initial block structure by scheduling the jobs in order of nondecreasing r_j; this requires $O(n)$ time. For each block B, select job $l \in B$ subject to (1), determine the block structure of the set $B - \{l\}$ by scheduling the jobs in this set in order of nondecreasing r_j, and construct the schedule for job l as given above; all this requires $O(|B|)$ time. By repeated application of this procedure to each of the subblocks, one obtains an optimal schedule in $O(n^2)$ time.

The algorithm generates at most $n - 1$ preemptions. This is easily proved by induction. It is obviously true for $n = 1$. Suppose it is true for blocks of size smaller than $|B|$. The schedule for block B contains at most $|B_i| - 1$ preemptions for each subblock B_i ($i = 1, \cdots, b$) and at most b preemptions for the selected job l, so that the total number of preemptions is no more than $\sum_{i=1}^{b} (|B_i| - 1) + b = |B| - 1$.

This bound on the number of preemptions is best possible. It is achieved by the class of problem instances defined by $r_j = -j$, $p_j = 2$, $f_j(t) = 0$ if $t \leq j$, $f_j(t) = 1$ otherwise ($j = 1, \cdots, n$). The only way to incur zero cost is to schedule job j in the intervals $[-j, -j + 1]$ and $[j - 1, j]$ ($j = 1, \cdots, n$). This uniquely optimal schedule contains $n - 1$ preemptions.

We note that the use of preemption is essential for our algorithm. If no preemption is allowed, it is not possible to determine the block structure of an optimal schedule in advance.

3. ARBITRARY RELEASE DATES, ARBITRARY PRECEDENCE CONSTRAINTS

We next consider the case in which arbitrary precedence constraints are specified. The algorithm can be extended to solve this more general problem as well.

First, the release dates are modified such that $r_j + p_j \le r_k$ whenever $j \to k$. This implies that, in constructing the blocks, one can ignore the precedence constraints. The modification requires $O(n^2)$ time: renumber the jobs in topological order (i.e., such that if $j \to k$ then $j < k$) and set $r_k = \max\{r_k, \max\{r_j + p_j | j \to k\}\}$ for $k = 2, \cdots, n$ (see Lageweg et al. [1976]). After this, the jobs are ordered according to nondecreasing r_j in $O(n \log n)$ time.

Secondly, suppose that for each job j belonging to a block B its *outdegree* d_j *with respect to* B, i.e. the number of jobs $k \in B$ such that $j \to k$, is known. For each block B, the set $B' = \{j | j \in B, d_j = 0\}$ of jobs without successors in B is determined, and the selection of job $l \in B$ subject to (1) is replaced by the selection of job $l \in B'$ such that $f_l(t(B)) = \min_{j \in B'}\{f_j(t(B))\}$. This ensures that the selected job l has no successors within the block B.

As to the implementation of this procedure, let us suppose that the precedence constraints are represented simply by a job adjacency matrix and the blocks by lists of jobs in order of nondecreasing r_j. Initially, for each job j the number d_j of jobs k such that $j \to k$ is recorded. The computation of all d_j requires $O(n^2)$ time. After construction of the initial blocks $B_1, \cdots, B_b$, the outdegrees are updated: consider each B_i ($i = 1, \cdots, b - 1$) in succession and decrease d_j by 1 for each pair (j, k) such that $j \in B_i$, $k \in \cup_{h=i+1}^{b} B_h$ and $j \to k$; d_j is then equal to the outdegree of job j with respect to the block to which it belongs. For each block B, after construction of the subblocks $B_1, \cdots, B_b$ of the set $B - \{l\}$ and of the schedule for job l, the outdegrees are updated again: consider each B_i ($i = 1, \cdots, b$) in succession and decrease d_j by 1 for each pair (j, k) such that $j \in B_i$, $k \in (\cup_{h=i+1}^{b} B_h) \cup \{l\}$ and $j \to k$. When a pair (j, k) is inspected, jobs j and k do not belong to the same block any more and hence the pair (j, k) will never be inspected again. It follows that each entry of the adjacency matrix is inspected at most once, so that all updates of the d_j together require $O(n^2)$ time. Further, each B' is determined in $O(|B|)$ time. We conclude that one still obtains an optimal schedule in $O(n^2)$ time.

TABLE I

PARAMETERS FOR THE EXAMPLE

	Job	j	1	2	3	4	5
(a)	release date	r_j	0	2	0	8	14
	processing time	p_j	4	2	4	2	4
(b)	modified release date	r_j	0	2	4	8	14
	outdegree	d_j	1	2	1	0	0
(c)	updated outdegree	d_j	1	2	0	0	
(d)	updated outdegree	d_j	0	0		0	

4. NUMERICAL EXAMPLE

Finally, we illustrate our algorithm with a numerical example. Suppose there are five jobs, with release dates and processing times as given in Table I(a), precedence constraints as specified in Figure 1, and cost functions as depicted in Figure 2.

The modified release dates and the initial outdegrees are given in Table I(b). By scheduling the jobs in order of increasing r_j, we obtain two blocks: $B_1 = \{1, 2, 3, 4\}$ from 0 until 12 and $B_2 = \{5\}$ from 14 until 18 (Figure 3(a)). We update the outdegrees of the jobs in B_1 as indicated in Table I(c).

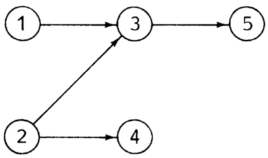

Figure 1. Precedence constraints for the example.

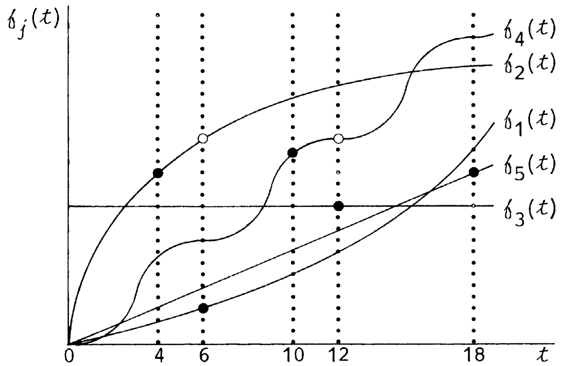

Figure 2. Cost functions for the example.

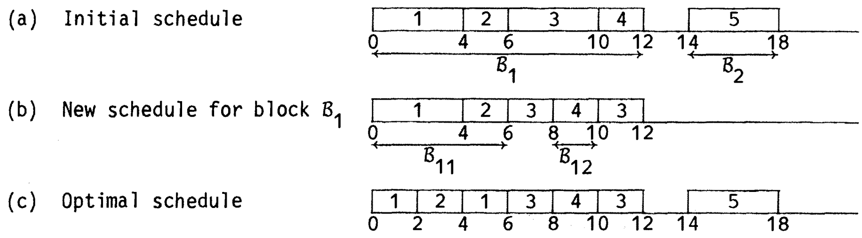

Figure 3. Schedules for the example.

Block B_2 consists of a single job and therefore represents an optimal part of the schedule. For block B_1 we find $B_1' = \{3, 4\}$ and select job 3 since $f_3(12) < f_4(12)$. By rescheduling the jobs in B_1 while processing job 3 only if no other job is available, we obtain two subblocks: $B_{11} = \{1, 2\}$ from 0 until 6 and $B_{12} = \{4\}$ from 8 until 10 (Figure 3(b)). We update the outdegrees of the jobs in B_{11} and B_{12} as indicated in Table I(d).

Block B_{12} needs no further attention. For block B_{11} we find $B_{11}' = \{1, 2\}$ and select job 1 since $f_1(6) < f_2(6)$. By rescheduling the jobs in B_{11} again, we finally obtain an optimal schedule (Figure 3(c)).

ACKNOWLEDGMENT

The authors gratefully acknowledge the useful comments from C. U. Martel. This research was partially supported by NSF Grant MCS78-20054.

REFERENCES

Garey, M. R., and D. S. Johnson. 1977. Two-Processor Scheduling with Start-Times and Deadlines. *SIAM J. Comput.* **6,** 416–426.

Graham, R. L., E. L. Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan. 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Ann. Discrete Math.* **5,** 287–326.

Lageweg, B. J., J. K. Lenstra and A. H. G. Rinnooy Kan. 1976. Minimizing Maximum Lateness on One Machine: Computational Experience and Some Applications. *Statist. Neerlandica* **30,** 25–41.

Lawler, E. L. 1973. Optimal Sequencing of a Single Machine Subject to Precedence Constraints. *Mgmt. Sci.* **19,** 544–546.

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