stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE
(DEPARTMENT OF OPERATIONS RESEARCH)

BW 112/79 SEPTEMBER

E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN

MINIMIZING MAXIMUM LATENESS IN A TWO-MACHINE OPEN SHOP

Preprint

2e boerhaavestraat 49 amsterdam

BIBLIOTHEEK MATHEMATISCH CENTRUM
AMSTERDAM

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

1980 Mathematics Subject Classification: 90B35, 68C15, 68C25

MINIMIZING MAXIMUM LATENESS IN A TWO-MACHINE OPEN SHOP

E.L. LAWLER
University of California, Berkeley

J.K. LENSTRA
Mathematisch Centrum, Amsterdam

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

ABSTRACT

We consider the problem of scheduling independent jobs in a two-machine open shop so as to minimize the maximum lateness with respect to due dates for the jobs. For the case in which preemption is allowed, a linear-time algorithm is presented. For the nonpreemptive case, NP-hardness is established.

KEY WORDS & PHRASES: *two-machine open shop, due dates, maximum lateness, preemptive scheduling, nonpreemptive scheduling, linear-time algorithm, NP-hardness.*

NOTE: This report is not for review; it has been submitted for publication in a journal.

1. INTRODUCTION

Consider the following open shop scheduling problem. There are n independent *jobs* $J_1,\ldots,J_n$ and two *machines* M_1,M_2. Each job J_j consists of two *operations*, one of length a_j which is to be executed on machine M_1 and one of length b_j which is to be executed on machine M_2. There is no restriction on the order in which the operations of a job are to be performed - hence the term *open shop*. Each job can be processed on at most one machine at a time, and either machine can process at most one operation at a time. No processing can occur prior to time zero.

A schedule is said to be *nonpreemptive* if each operation is executed continuously from start to completion. A schedule is *preemptive* if the execution of any operation may arbitrarily often be interrupted and resumed at a later time; the periods in which the operations of a given job are performed may be interleaved in time.

Each schedule defines a *completion time* C_j for each J_j. Given a *due date* d_j for each J_j, we define its *lateness* $L_j = C_j - d_j$. Common optimality criteria involve the minimization of the *maximum completion time* $C_{max} = \max_{1\le j\le n}\{C_j\}$ or the *maximum lateness* $L_{max} = \max_{1\le j\le n}\{L_j\}$.

In this paper we shall be concerned with the minimization of L_{max} in a two-machine open shop. We present a linear-time algorithm for the preemptive case and, by contrast, we establish NP-hardness for the nonpreemptive case. Our algorithm presupposes that the jobs are ordered according to nondecreasing due dates; if this is not the case, its running time would be $O(n \log n)$ rather than $O(n)$. An $O(n^2)$ algorithm for this problem was obtained in [2]. Our NP-hardness result is "strong" in the sense that it holds even with respect to a unary encoding of the problem data [4]. It should be apparent that both results also apply to the minimization of C_{max} in a two-machine open shop subject to arbitrary *release dates* for the jobs.

In the literature on open shop scheduling, most attention has been paid to the minimization of C_{max} in the absence of release dates and due dates. In the case of two machines, there is no advantage to preemption, and there exists a linear-time algorithm to find an optimal nonpreemptive schedule [6;7]. In the case of m machines, the preemptive problem can be solved in polynomial time for arbitrary m [5;6;8], whereas the nonpreemptive

problem is binary NP-hard for any fixed $m \geq 3$ [6] and unary NP-hard for arbitrary m [7].

The most general preemptive problem in this context is the minimization of L_{max} in an m-machine open shop subject to release dates. This problem can be formulated as a linear program [2]. Thus, the recent development of a polynomial-time algorithm for linear programming [1;3] is of interest to the area of open shop scheduling as well.

2. THE PREEMPTIVE CASE

We shall first describe a procedure to determine the existence of a schedule with value $L_{max} \leq 0$; that is, we will view the due dates as absolute *deadlines* and test these deadlines for feasibility. We shall then modify this procedure to compute the minimum value of L_{max}. We shall also show how to construct a schedule with that value. Finally, we shall determine the maximum number of preemptions introduced into such a schedule.

Testing for feasibility

We assume that the jobs are indexed according to their due dates, i.e., $d_1 \leq \ldots \leq d_n$, and define $A_j = \sum_{k=1}^{j} a_k$, $B_j = \sum_{k=1}^{j} b_k$ $(j = 1,\ldots,n)$. We now view the due dates as absolute deadlines and ask whether or not there exists a feasible schedule.

The jobs are scheduled in order of nondecreasing deadlines. Suppose that $J_1,\ldots,J_{j-1}$ have been successfully scheduled and that J_j has to be scheduled next. Let x_j (y_j) denote the total amount of time prior to d_j that M_1 (M_2) is idle while M_2 (M_1) is busy, and let z_j denote the total amount of time prior to d_j that M_1 and M_2 are simultaneously idle. Note that x_j, y_j, z_j are not independent, inasmuch as

$$x_j + z_j = d_j - A_{j-1}, \tag{1}$$

$$y_j + z_j = d_j - B_{j-1}. \tag{2}$$

The minimum amount of processing of the operation of J_j on M_1 (M_2) that must be done while both machines are available is $\max\{0, a_j - x_j\}$ $(\max\{0, b_j - y_j\})$. It follows that J_j can be successfully scheduled if and only if

$$\max\{0, a_j - x_j\} + \max\{0, b_j - y_j\} \leq z_j.$$

This inequality is equivalent to the four inequalities

$$0 \leq z_j,$$

$$a_j - x_j \leq z_j,$$

$$b_j - y_j \leq z_j,$$
$$a_j - x_j + b_j - y_j \leq z_j.$$

Discarding the first of these inequalities as vacuous and applying (1) and (2) to the remaining ones, we see that J_j can be successfully scheduled if and only if each of the following feasibility conditions holds:

$$A_j \leq d_j, \tag{3}$$
$$B_j \leq d_j, \tag{4}$$
$$A_j + B_j \leq 2d_j - z_j. \tag{5}$$

These inequalities also tell us that in order to obtain a feasible schedule we should attempt to minimize the value of z_j at each iteration. It is easily seen that the smallest possible values of $z_1,\ldots,z_n$ are defined recursively by

$$z_1 = d_1, \quad z_j = d_j - d_{j-1} + \max\{0, z_{j-1} - a_{j-1} - b_{j-1}\} \quad (j = 2,\ldots,n). \tag{6}$$

We have thus arrived at an O(n) procedure to determine the existence of a feasible schedule: for $j = 1,\ldots,n$, compute z_j by (6) and test (3), (4) and (5). There exists a feasible schedule if and only if all these tests succeed.

Minimizing maximum lateness

We propose to determine the minimum value of L_{max} by carrying out a parametrized version of the preceding computation. Each d_j is replaced by $d_j + L$, where L is a free parameter. The smallest value of L for which there exists a feasible schedule with respect to the deadlines $d_j + L$ is evidently equal to the minimum value of L_{max} that can be achieved with respect to the original due dates d_j.

Let us first rewrite the z_j, as defined by (6), for deadlines $d_j + L$ in such a way that L does not appear in the recursive part of the expression. We claim that

$$z_j = \max\{d_j + L - A_{j-1} - B_{j-1}, z'_j\} \quad (j = 1,\ldots,n), \tag{7}$$

where $A_0 = B_0 = 0$ and

$$z'_1 = -\infty, \quad z'_j = d_j - d_{j-1} + \max\{0, z'_{j-1} - a_{j-1} - b_{j-1}\} \quad (j = 2,\ldots,n). \tag{8}$$

This is easily proved inductively: (7) is clearly true for j = 1, and assuming (7) is true for j-1, we get

$$\begin{aligned} z_j &= (d_j+L) - (d_{j-1}+L) + \max\{0, \max\{d_{j-1}+L-A_{j-2}-B_{j-2}, z'_{j-1}\} - a_{j-1} - b_{j-1}\} \\ &= \max\{d_j+L-A_{j-1}-B_{j-1}, z'_j\}. \end{aligned}$$

By substituting (7) into (5), we can write the feasibility conditions (3), (4) and (5) for deadlines d_j+L as follows:

$$\begin{aligned} A_j &\le d_j+L, \\ B_j &\le d_j+L, \\ a_j+b_j &\le d_j+L, \\ A_j+B_j &\le 2(d_j+L) - z'_j. \end{aligned}$$

The smallest value of L for which these inequalities are satisfied for j = 1,...,n is given by

$$L^* = \max_j\{\max\{A_j, B_j, a_j+b_j, \tfrac{1}{2}(A_j+B_j+z'_j)\} - d_j\}. \tag{9}$$

We have thus obtained an O(n) procedure to determine the minimum value of L_{max}: for j = 1,...,n, compute z'_j by (8), and compute L^* by (9).

We note that, in the case that all $d_j = 0$, (9) reduces to

$$L^* = \max\{A_n, B_n, \max_j\{a_j+b_j\}\},$$

which is the minimum value of C_{max} as given in [6;7].

Constructing an optimal schedule

Suppose we have determined L^* as above and we now wish to actually construct a schedule with that value, i.e., a feasible schedule with respect to deadlines d_j+L^*. We will show how this can be accomplished in linear time as well.

At the time that J_j is to be scheduled, the available idle time on the two machines has the following structure (cf. Figure 1). There are various intervals during which M_1 is idle but M_2 is busy, and other intervals in

which the reverse is true; these intervals have total lengths x_j and y_j respectively. The last of them is of the latter type and is denoted by Y_j. This interval is immediately followed by a single interval Z_j of length z_j, just prior to d_j+L^*, during which both M_1 and M_2 are idle.

We schedule J_j in such a way that this structure is preserved at the time that J_{j+1} is to be scheduled. Recall that we must utilize as much as possible of the interval Z_j in order to minimize z_{j+1}. This period can be used in its entirety if and only if $a_j+b_j \geq z_j$. If $a_j+b_j < z_j$, we schedule J_j as indicated schematically in Figure 2. If $a_j+b_j \geq z_j$, we attempt to perform as much as possible of the operation of J_j on M_1 during Z_j. The maximum amount of this operation that can be performed during Z_j is given by

$$a_j' = \min\{a_j, z_j, z_j-(b_j-y_j)\}.$$

In each of the three cases $a_j' = a_j$, $a_j' = z_j$, $a_j' = z_j-(b_j-y_j)$, we schedule J_j as indicated schematically in Figure 3. The reader should verify that the resulting idle time structure facing J_{j+1} satisfies the above description.

All the intervals during which either M_1 or M_2 is available, except Y_j, can be maintained in a LIFO stack. It should be apparent how the necessary operations can be implemented without further comment. The analysis of the number of preemptions below will confirm that our schedule construction procedure requires O(n) time.

The number of preemptions

In order to analyze the maximum number of preemptions created, we introduce the notion of an active period (cf. [8]). An *active period* is a maximal interval of time during which a machine continuously performs the same operation. The number of preemptions in a schedule is equal to the number of active periods in excess of the number of operations.

Consider the situation immediately after J_j has been scheduled. Let there be ρ_j active periods and σ_j intervals in the two stacks, and let $\tau_j = \rho_j+\sigma_j$. Examination of our schedule construction procedure shows that

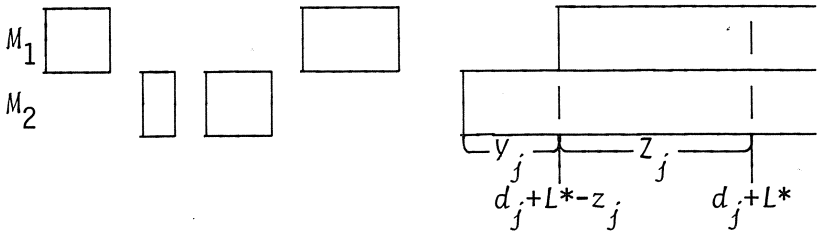


Figure 1 Typical arrangement of idle intervals.

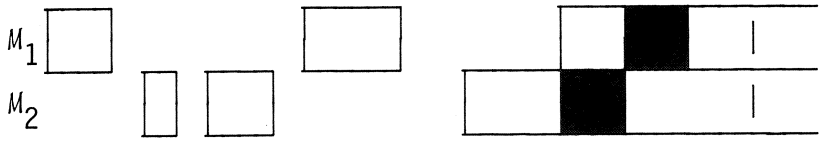
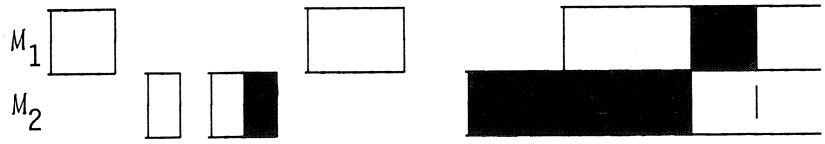
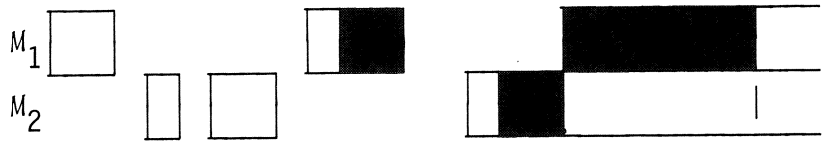


Figure 2 Schedule for J_j if $a_j+b_j < z_j$.



(a) $a'_j = a_j$.



(b) $a'_j = z_j$.

(c) $a'_j = z_j-(b_j-y_j)$.

Figure 3 Schedule for J_j if $a_j+b_j \geq z_j$.

$$\tau_1 \leq 3, \quad \tau_j \leq \tau_{j-1}+4 \quad (j > 1),$$

so that $\tau_j \leq 4j-1$. It is also easily seen that

$$\rho_1 \leq 2, \quad \rho_j \leq \rho_{j-1}+\sigma_{j-1}+2 = \tau_{j-1}+2 \quad (j > 1),$$

and hence $\rho_j \leq 4j-3$ for $j > 1$.

It follows that the number of preemptions introduced into an optimal schedule cannot exceed $\rho_n - 2n \leq 2n-3$ for $n > 1$. It should be evident from this result that our schedule construction procedure requires $O(n)$ time.

Although our algorithm can actually produce schedules with $2n-3$ preemtions, it remains an open question whether this bound is tight. We have been unable to find problem instances that require more than $2n-5$ preemptions.

Comments

It has already been observed that our algorithm can also be applied to minimize C_{max} subject to release dates for the jobs. Moreover, our techniques can be adapted to determine the existence of a feasible schedule with respect to both release dates r_j and deadlines d_j, provided that the intervals $[r_j,d_j]$ are *nested* (i.e., for all j,k, $[r_j,d_j] \cap [r_k,d_k] \in \{\emptyset, [r_j,d_j], [r_k,d_k]\}$). This problem is solved by working from the innermost intervals in the nesting outward; we leave details to the reader.

Our analysis allows no direct extension to the minimization of L_{max} on three machines. In that case, the situation just before J_j is to be scheduled cannot be characterized by a single variable z_j and an optimal scheduling rule is not obvious at all. Nevertheless, the linear programming formulation mentioned above proves that this problem is solvable in polynomial time.

A related two-machine open shop problem involves the minimization of the *weighted number of late jobs*. By means of relatively straightforward dynamic programming techniques, it is possible to solve this problem by a pseudopolynomial algorithm in $O(nd_n^3)$ time, provided that the data a_j, b_j, d_j are integers.

We note that in preemptive scheduling it is often expedient to consider

the intervals between adjacent release dates or deadlines in succession. In this paper, we have scheduled the jobs at each iteration in such a way that the distribution of the remaining machine capacities carried over to the next iteration is optimal. An alternative approach would be to schedule the jobs such that the distribution of their remaining execution requirements is always as favorable as possible. Perhaps the latter strategy would have to be followed in order to obtain similar results for a larger class of problems.

3. THE NONPREEMTIVE CASE

In this section we complement the result of the previous section by establishing NP-hardness for the minimization of L_{max} in a two-machine open shop when no preemptions are allowed. This result will be obtained by specifying a polynomial transformation from the following NP-complete problem [4]:

> 3-PARTITION: Given a set $S = \{1,\ldots,3t\}$ and positive integers $p_1,\ldots,p_{3t},q$ with $\frac{q}{4} < p_j < \frac{q}{2}$ for all $j \in S$ and $\sum_{j\in S} p_j = tq$, does S have a partition into t disjoint 3-element subsets $S_1,\ldots,S_t$ such that $\sum_{j\in S_i} p_j = q$ for $i = 1,\ldots,t$?

Given any instance of 3-PARTITION, we define an instance of the two-machine open shop problem as follows:

$n = 4t;$

$a_j = 0,\ b_j = p_j,\ d_j = tq+t \quad (j \in S);$

$a_{3t+1} = 0,\ b_{3t+1} = 1,\ d_{3t+1} = 1;$

$a_{3t+i} = q+1,\ b_{3t+i} = 1,\ d_{3t+i} = (i-1)q+i \quad (i = 2,\ldots,t).$

We claim that 3-PARTITION has a solution if and only if there exists a nonpreemptive schedule with value $L_{max} \leq 0$, i.e., in which no job is late.

Let us first investigate when the jobs J_{3t+i} $(i = 1,\ldots,t)$ can be executed in such a schedule. Clearly, J_{3t+1} has to be processed on M_2 during the interval $[0,1]$. In order for J_{3t+2} to meet its due date, it has to occupy M_1 during the interval $[0,q+1]$ and M_2 during the interval $[q+1,q+2]$. A straightforward inductive argument shows that J_{3t+i} has to be executed on M_1 during $[(i-2)q+i-2,(i-1)q+i-1]$ and on M_2 during $[(i-1)q+i-1,(i-1)q+i]$, for $i = 2,\ldots,t$ (cf. Figure 4).

This leaves t intervals $[(i-1)q+i,iq+i]$ $(i = 1,\ldots,t)$, each of length q, for the execution of the jobs J_j $(j \in S)$ on M_2. It follows that there exists a schedule with value $L_{max} \leq 0$ if and only if the jobs J_j $(j \in S)$ can be divided into t groups, each containing 3 jobs and requiring q units of processing time on M_2, i.e., if and only if 3-PARTITION has a solution.

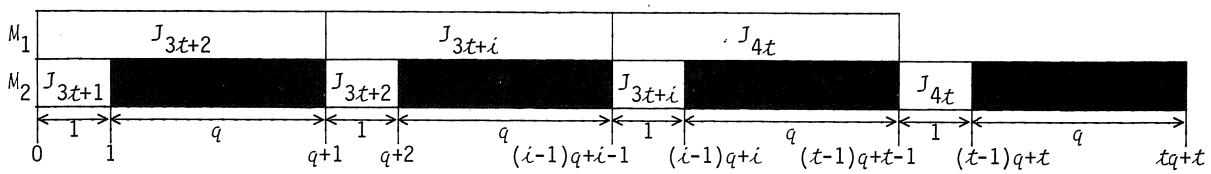


Figure 4 Illustration of the transformation.

ACKNOWLEDGMENTS

This research was partially supported by NSF Grant MCS76-17605 and by NATO Special Research Grant 9.2.02 (SRG.7).

REFERENCES

1. L.G. CHAČIJAN (1979) Polynomial'nyĭ algoritm v lineĭnom programmirovanii (A polynomial algorithm for linear programming; in Russian). *Doklady Akad. Nauk USSR* 244,1093-1096.
2. Y. CHO, S. SAHNI (1978) Preemptive scheduling of independent jobs with release and due times on open, flow and job shops. Technical Report 78-5, Computer Science Department, University of Minnesota, Minneapolis.
3. P. GÁCS, L. LOVÁSZ (1979) Khachian's algorithm for linear programming. Computer Science Department, Stanford University, California.
4. M.R. GAREY, D.S. JOHNSON (1979) *Computers and Intractability: a Guide to the Theory of NP-Completeness,* Freeman, San Francisco.
5. T. GONZALEZ (1976) A note on open shop preemptive schedules. Technical Report 214, Computer Science Department, Pennsylvania State University.
6. T. GONZALEZ, S. SAHNI (1976) Open shop scheduling to minimize finish time. *J. Assoc. Comput. Mach.* 23,665-679.
7. R.L. GRAHAM, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.* 5,287-326.
8. E.L. LAWLER, J. LABETOULLE (1978) On preemptive scheduling of unrelated parallel processors by linear programming. *J. Assoc. Comput. Mach.* 25,612-619.