stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE
(DEPARTMENT OF OPERATIONS RESEARCH)

BW 105/79 MEI

E.L. LAWLER

PREEMPTIVE SCHEDULING OF UNIFORM PARALLEL MACHINES TO MINIMIZE THE WEIGHTED NUMBER OF LATE JOBS

Preprint

2e boerhaavestraat 49 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).

AMS subject classification scheme (1979): 90B35, 90C39, 68C05, 68C15, 68C25

PREEMPTIVE SCHEDULING OF UNIFORM PARALLEL MACHINES TO MINIMIZE THE WEIGHTED NUMBER OF LATE JOBS

E.L. LAWLER

Computer Science Division, University of California, Berkeley, CA 94720, USA

ABSTRACT

We show that it is possible to preemptively schedule n jobs on m uniform parallel machines so as to minimize the weighted number of late jobs in time of $O(W^2n^2)$, for m = 2, and of $O(W^2n^{3m-5})$, for $m \geq 3$, where W is the sum of the integer weights of the jobs. For fixed m this constitutes a pseudopolynomial time bound, and for fixed m and unweighted jobs, i.e. W = n, a strictly polynomial time bound. It is also shown that for fixed m there is a fully polynomial approximation scheme.

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

KEY WORDS & PHRASES: *preemptive scheduling, parallel machines, dynamic programming, polynomial algorithm, pseudopolynomial algorithm, NP-hardness, fully polynomial approximation scheme.*

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

1. INTRODUCTION

One possible objective in scheduling jobs for processing within a given machine environment is to minimize the (weighted or unweighted) number of jobs which are late with respect to specified due dates. In this paper we deal with this objective in the context of preemptive scheduling of "uniform" parallel machines. We show that for any fixed number of machines it is possible to obtain an optimal schedule for unweighted jobs in polynomial time and for weighted jobs in pseudopolynomial time. We also obtain a fully polynomial approximation scheme for the latter case.

These results can be compared with known results on nonpreemptive scheduling of parallel machines with respect to the same objective. The unweighted single machine problem can be solved by a procedure due to Moore in $O(n \log n)$ time, where n is the number of jobs [4,7]. The weighted single machine problem is NP-hard, but can be solved by dynamic programming in $O(Wn)$ time, where W is the sum of the integer job weights [6]. The unweighted problem is well known to be NP-hard for even two identical machines [2]. However, dynamic programming can be applied to solve the weighted problem in $O(nP^m)$ time for m uniform parallel machines, where P is the sum of the integer job processing requirements [2,8]. For any fixed number of machines, this constitutes a pseudopolynomial time solution procedure.

It is not difficult to demonstrate that in the case of a single machine there is no advantage to be gained from preemption. That is, for any preemptive schedule, there is a nonpreemptive schedule which is at least as good. Thus, the $O(n \log n)$ and $O(Wn)$ procedures cited above also apply to preemptive scheduling of a single machine. Moreover, the NP-hardness result concerning the weighted nonpreemptive scheduling of a single machine also applies to the preemptive scheduling of a single machine and, a fortiori, to the preemptive scheduling of any number of uniform parallel machines.

The results presented in this paper are obtained by the application of dynamic programming techniques to preemptive scheduling procedures derived from those of GONZALEZ and SAHNI [1] and SAHNI and CHO [9]. We are able to solve the weighted scheduling problem in pseudopolynomial time for any fixed number of uniform parallel machines: $O(W^2n^2)$ for $m = 2$, and $O(W^2n^{3m-5})$ for $m \geq 3$. When the jobs are unweighted, i.e. $W = n$, these running times are

strictly polynomial. A fully polynomial approximation scheme is also obtained in which n^2/ε replaces W^2 in each of the running times cited above for the weighted problem.

2. DEFINITIONS

We make the usual assumptions of parallel scheduling: a machine can process at most one job at a time and a job can be processed by at most one machine at a time. The schedules we consider are preemptive, in that the processing of a job can be interrupted at any time t and processing resumed at any time $t' \geq t$ on the same machine, or a different machine, without penalty.

We find it convenient to generalize the usual definition of uniform parallel machines so as to allow the speeds of the machines to be time varying. Let $s_i(t)$, $i = 1,2,\ldots,m$, denote the instantaneous speed of machine i at time t and assume, for all t, that $s_1(t) \geq s_2(t) \geq \ldots \geq s_m(t)$. The *processing capacity* of machine i in the time interval $[t,t']$ is

$$\int_t^{t'} s_i(u)du.$$

Each job j, $j = 1,2,\ldots,n$, has a specified *processing requirement* $p_j > 0$. In order for a job to be completed, it is necessary that the sum of the processing capacities in the time intervals in which the job is processed should equal its processing requirement. For example, if job j is processed on machine 1 in interval $[t_1,t_1']$ and on machine 2 in interval $[t_2,t_2']$, then this processing is sufficient to complete the job if

$$\int_{t_1}^{t_1'} s_1(u)du + \int_{t_2}^{t_2'} s_2(u)du = p_j.$$

In addition to machine speeds and job processing requirements, a *due date* $d_j > 0$ and a *weight* $w_j > 0$ are specified for each job j. We assume that job weights are integers. However, no such assumption is made about processing requirements and due dates.

With respect to a given feasible schedule, a job is *on time* if its processing is completed by its due date and *late* otherwise. Our objective is to minimize the sum of the weights of the jobs which are late. Thus our problem is equivalent to that of finding a subset of jobs of maximum total weight such that there exists a schedule in which all the jobs in the subset are completed on time.

3. SCHEDULING JOBS WITHIN A FIXED TIME INTERVAL

Let us first consider the problem of constructing a feasible schedule for n given jobs within a specified time interval. The procedure we present is a modification of a procedure due to GONZALEZ and SAHNI [1]. (A different type of procedure is described in [3,10].)

Let S_i, $i = 1,2,\ldots,m$, denote the processing capacity of machine i in the interval $[0,T]$. Assume $p_1 \geq p_2 \geq \ldots \geq p_n$. In order for there to exist a feasible preemptive schedule within the interval $[0,T]$ it is necessary that

$$\begin{aligned} S_1 &\geq p_1 \\ S_1+S_2 &\geq p_1+p_2 \\ &\vdots \\ S_1+S_2+\ldots+S_{m-1} &\geq p_1+p_2+\ldots+p_{m-1} \\ S_1+S_2+\ldots+S_{m-1}+S_m &\geq p_1+p_2+\ldots+p_n. \end{aligned} \tag{3.1}$$

We assert that not only are conditions (3.1) necessary for the existence of a feasible schedule, but they are sufficient as well. We shall prove sufficiency of conditions (3.1) by describing a procedure for actually constructing a schedule within the interval $[0,T]$.

The procedure schedules the jobs one at a time, in arbitrary order. Let us suppose we first choose to schedule job j. We find the machine with largest index k such that $S_k \geq p_j$. There are three possible cases:

Case 1: $S_k = p_j$. In this case we simply schedule job j to be processed by machine k for the entire period $[0,T]$. We then eliminate job j and machine k from the problem, leaving a problem with n-1 jobs and m-1 machines.

Case 2: $k \leq m-1$, $S_k > p_j > S_{k+1}$. We assert that there exists a time t', $0 < t' < T$, such that

$$p_j = \int_0^{t'} s_k(u)\,du + \int_{t'}^{T} s_{k+1}(u)\,du.$$

To convince ourselves of this fact, we need only consider the plot of the curves for

$$f(t) = \int_0^t s_k(u)du, \quad g(t) = \int_t^T s_{k+1}(u)du,$$

and $f(t)+g(t)$, as shown in Figure 1.

It is apparent that there must be at least one point t', $0 < t' < T$, such that $f(t')+g(t') = p_j$. This is guaranteed by the facts that $f(T) = S_k > p_j > S_{k+1} = g(0)$, $f(0) = g(T) = 0$, and that f and g are continuous functions.

We now propose to schedule job j for processing on machine k in the interval $[0,T]$ and on machine k+1 in the interval $[t',T]$. We then create a *composite* machine from the remaining available time on the "elementary" machines k and k+1. This new composite machine has speed $s_{k+1}(t)$ in the interval $[0,t']$ and speed $s_k(t)$ in the interval $[t',t]$. The capacity of this composite machine in the interval $[0,T]$ is thus

$$S_k + S_{k+1} - p_j = \int_0^{t'} s_{k+1}(u)du + \int_{t'}^T s_k(u)du.$$

We then replace the elementary machines k and k+1 by the new composite machine and eliminate job j from the problem, leaving a problem with n-1 jobs and m-1 machines.

Case 3: $S_m > p_j$. In this case we simply schedule job j in the available time on machine m, thereby reducing the capacity of that machine to $S_m - p_j$. The problem is thus reduced to one involving n-1 jobs and m machines.

It is seen that in each of the three cases the problem is reduced to one of the same type involving machines with time-varying speeds, but with only n-1 jobs. It is possible to verify that conditions (3.1) are satisfied for each of these smaller problems, and we leave this as an exercise for the reader. We thus can obtain a proof that repeated application of this procedure yields a feasible schedule of the n jobs within the time interval $[0,T]$. (It is also possible to show that the feasible schedule constructed contains no more than 2(m-1) preemptions.)

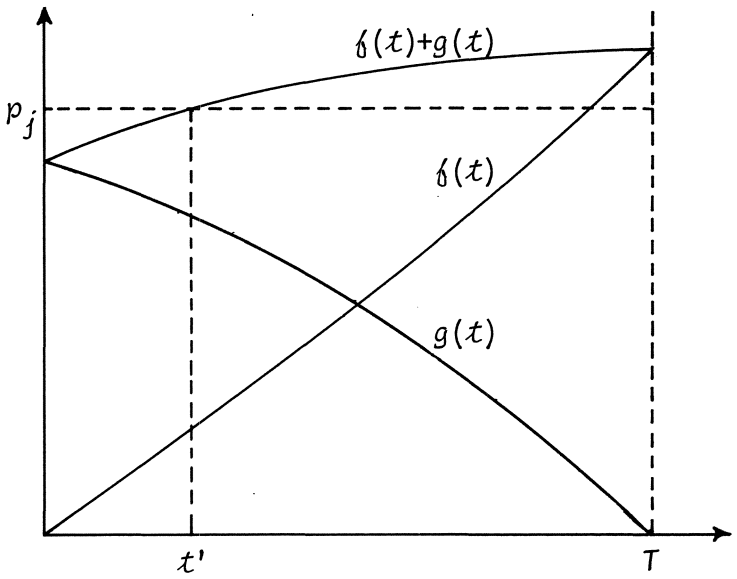


Figure 1 Plots of f(t), g(t), f(t)+g(t).

4. SCHEDULING JOBS SUBJECT TO DEADLINES

Now let us consider the problem of scheduling n jobs, subject to absolute deadlines d_j, $j = 1,2,\ldots,n$, on the times at which the jobs must be completed. We shall present a modified version of a procedure due to SAHNI CHO [8].

Assume, without loss of generality, that $0 \le d_1 \le d_2 \le \ldots \le d_n$. We try to schedule the jobs one by one, in deadline order, from earliest to latest. In the interval $[0,d_1]$, the m machines have processing capacities

$$s_i^{(1)} = \int_0^{d_1} s_i(u)du, \quad i = 1,2,\ldots,m.$$

If job 1 is to be completed on time, it must be scheduled within the interval $[0,d_1]$. Hence we apply the procedure described in the previous section, obtaining remaining composite processing capacities $\bar{s}_i^{(1)}$, $i = 1,2,\ldots,m$, where

$$\begin{aligned}
\bar{s}_i^{(1)} &= s_i^{(1)}, && i \le k-1,\\
\bar{s}_k^{(1)} &= s_k^{(1)} + s_{k+1}^{(1)} - p_j,\\
\bar{s}_i^{(1)} &= s_{i+1}^{(1)}, && k+1 \le i \le m-1,\\
\bar{s}_m^{(1)} &= 0,
\end{aligned}$$

if Cases 1 or 2 apply, and $\bar{s}_i^{(1)} = s_i^{(1)}$, $1 \le i \le m-1$, $\bar{s}_m^{(1)} = s_m^{(1)} - p_1$, if Case 3 applies.

For the time interval $[0,d_2]$, we now have composite machines with capacities

$$s_i^{(2)} = \bar{s}_i^{(1)} + \int_{d_1}^{d_2} s_i(u)du.$$

If job 2 is to be completed on time, it must be scheduled within the interval $[0,d_2]$. Hence we again apply the procedure described in the previous section, obtaining remaining composite processing capacities $\bar{s}_i^{(2)}$, $i = 1,2,\ldots,m$.

We continue in this way until either job n has been scheduled, or until for some j we find that $p_j > s_1^{(j)}$. We assert that in this latter case no feasible schedule exists.

It is not difficult to prove that this procedure yields a feasible schedule if one exists. Note that if a feasible schedule does exist, then processing requirements $p'_1 = p_1$, $p'_j \leq p_j$, $j = 2,\ldots,n$, are processed within the interval $[0,d_1]$. Since it is possible to schedule n jobs with requirements p'_j, $j = 1,2,\ldots,n$, within $[0,d_1]$, it is certainly possible to schedule them with job 1 scheduled as we did, by the arguments of the previous section. This observation suggests the form of an inductive proof, the details of which we leave to the reader.

5. A NUMERICAL EXAMPLE

As a simple numerical example, consider a problem with four uniform processors, with constant speeds $s_1 = 4$, $s_2 = 3$, $s_3 = 2$, $s_1 = 1$, i.e. $S_i^{[t,t']} = s_i(t'-t)$, and with seven jobs with deadlines and processing requirements as follows:

j	p_j	d_j
1	1	1
2	6	2
3	1	3
4	11	4
5	2	5
6	1	6
7	20	7

Let $S_i^{(j)}$ denote the capacity of composite processor i at time d_j before job j is scheduled, and $\bar{S}_i^{(j)}$ the capacity of processor i at time d_j after job j is scheduled. Then, applying the procedure of the previous section, we obtain the following composite processor capacities:

i	$S_i^{(1)}$	$\bar{S}_i^{(1)}$	$S_i^{(2)}$	$\bar{S}_i^{(2)}$	$S_i^{(3)}$	$\bar{S}_i^{(3)}$	$S_i^{(4)}$	$\bar{S}_i^{(4)}$	$S_i^{(5)}$	$\bar{S}_i^{(5)}$	$S_i^{(6)}$	$\bar{S}_i^{(6)}$	$S_i^{(7)}$	$\bar{S}_i^{(7)}$
1	4	4	8	8	12	12	16	15	19	19	23	23	27	21
2	3	3	6	4	7	7	10	5	8	8	11	11	14	6
3	2	2	4	1	3	3	5	1	3	2	4	4	6	1
4	1	0	1	0	1	0	1	0	1	0	1	0	1	0

The actual schedule constructed is indicated by the Gannt chart in Figure 2, in which the shaded areas indicate machine idle time.

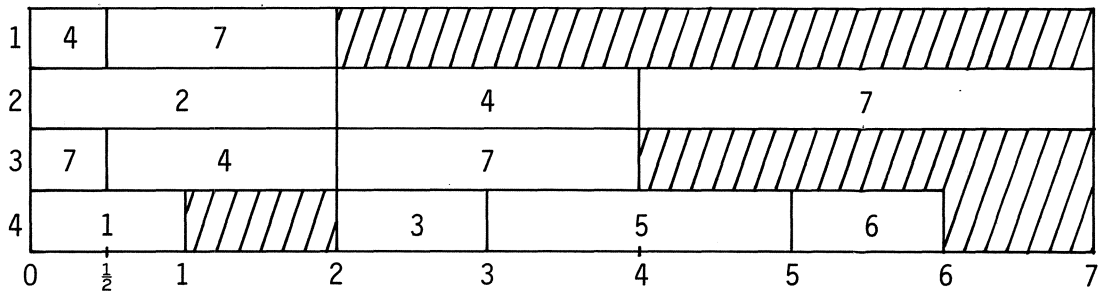
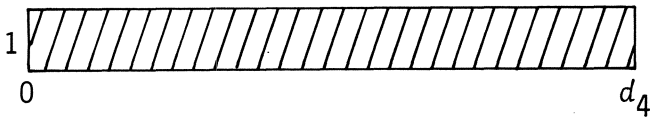
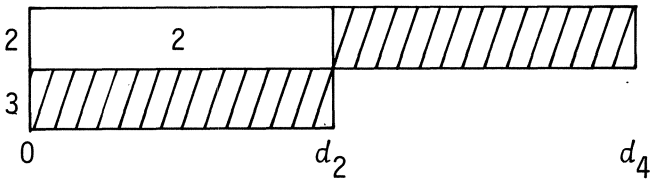


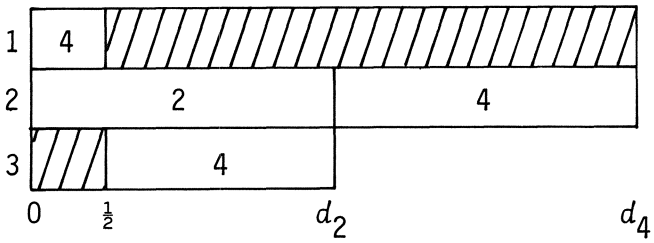
Figure 2 Schedule obtained for example.



(a) Composite machine 1, prior to scheduling job 4.



(b) Composite machine 2, prior to scheduling job 4.



(c) Composite machine 1, after scheduling job 4.

Figure 3

6. THE "SHAPE" OF A COMPOSITE MACHINE

We wish to define the "shape" of a composite machine and the set of jobs which are "charged" to it. In order to motivate these concepts, we shall first consider in detail how some of the $S_i^{(j)}$ and $\bar{S}_i^{(j)}$ values were obtained in the numerical example of the previous section.

Just prior to the scheduling of job 4, composite machine 1 is composed simply of elementary machine 1 in the interval $[0,d_4]$ and we have

$$S_1^{(4)} = s_1(d_4-0) = 16. \tag{6.1}$$

Composite machine 1 is represented by the simple diagram in Figure 3(a).

Also just prior to the scheduling of job 4, composite machine 2 is composed of elementary machine 3 in the interval $[0,d_2]$ and elementary machine 2 in the interval $[d_2,d_4]$, from which it follows that

$$S_2^{(4)} = s_3(d_2-0) + s_2(d_4-d_2) = 10.$$

The reason that elementary machine 3 is part of the composed machine 2 in the interval $[0,d_2]$ is that job 2 was scheduled on machine 2 in the interval, making

$$\begin{aligned}\bar{S}_2^{(2)} &= S_2^{(2)} + S_3^{(2)} - p_2 \\ &= s_2(d_2-0) + s_3(d_2-0) - p_2,\end{aligned}$$

from which it follows that

$$\begin{aligned}S_2^{(4)} &= \bar{S}_2^{(2)} + s_2(d_4-d_2) \\ &= s_2(d_4-0) + s_3(d_2-0) - p_2.\end{aligned} \tag{6.2}$$

Composite machine 2 is represented by the diagram in Figure 3(b).

After job 4 is scheduled, a new composite machine 1 is obtained, with

$$\begin{aligned}\bar{S}_1^{(4)} &= S_1^{(4)} + S_2^{(4)} - p_2 \\ &= s_1(d_4-0) + s_2(d_4-0) + s_3(d_2-0) - (p_2+p_4).\end{aligned} \tag{6.3}$$

The new composite machine 1 is represented by the diagram in Figure 3(c).

These examples suggest that for each composite machine at each point in time there is a well-defined set of time intervals on elementary machines corresponding to those indicated in the diagrams in Figure 3. These time intervals will be said to determine the *shape* of the composite machine. The shape of a composite machine together with the subset of jobs whose processing requirements are *charged* to that machine, determines the processing capacity of the machine, as in equations (6.1)-(6.3).

Initially, at time d_1 prior to scheduling job 1, the shape of composite machine i is specified by the interval $[0,d_1]$ on elementary machine i. The set of jobs J_i charged to composite machine i is empty. Thereafter, as the computation proceeds, the shape of each composite machine and the set of jobs charged to it is easily determined. For example, suppose that at time d_j job j is scheduled on composite machines k and k+1 so that

$$\bar{s}_k^{(j)} = s_k^{(j)} + s_{k+1}^{(j)} - p_j.$$

Then the shape of the new composite machine k is the union of the shapes of the previous machines k and k+1 and

$$J_k := J_k \cup J_{k+1} \cup \{j\}.$$

In this case, we have

$$\bar{s}_i^{(j)} = s_{i+1}^{(j)}, \quad k < i < m.$$

Accordingly, the shape of each new composite machine i, is the shape of the previous machine i+1 and $J_i := J_{i+1}$. We also have

$$\bar{s}_m^{(j)} = 0.$$

Accordingly, the shape of the new composite machine m is empty (or, equivalently, consists of the interval $[d_j,d_j]$ on elementary machine m) and $J_m := \emptyset$.

Corresponding to the relations

$$s_i^{(j+1)} = \bar{s}_i^{(j)} + \int_{d_j}^{d_{j+1}} s_i(u)du,$$

the shape of composite machine i is augmented by the interval $[d_j,d_{j+1}]$ on

elementary machine i in moving from time d_j to time d_{j+1}. The set of jobs charged to the composite machine of course remains unchanged.

A convenient way to specify the shape of composite machine i is by two sets

$$X = \{x(i), x(i+1), \ldots, x(g)\} \subseteq \{1,2,\ldots,n\}$$

and

$$Y = \{y(i+1), y(i+2), \ldots, y(n)\} \subseteq \{1,2,\ldots,n\},$$

where $g \leq h \leq m$. Assuming $x(i) > x(i+1) > \ldots > x(g)$ and $y(i+2) > y(i+2) > \ldots > y(h)$, the shape X,Y is indicated by Figure 4. If J_i is the set of jobs charged to the composite machine, then

$$S_i^{(j)} = \int_{d_{x(i)}}^{d_j} s_i(u)du + \sum_{k=i+1}^{g} \int_{d_{x(k)}}^{d_{y(k)}} s_k(u)du + \sum_{k=g+1}^{h} \int_{0}^{d_{y(k)}} s_k(u)du - \sum_{j \in J_i} p_j.$$

(If $X = \emptyset$, then the first integration is from 0 to d_j).

Let X_i, Y_i be the shape of composite machine i in this notation. Initially, at time d_1 prior to scheduling job 1,

$$X_i := \emptyset, \; Y_i := \emptyset, \quad 1 \leq i \leq m.$$

Suppose job j is scheduled on composite machines k and k+1. We assert that it is necessarily the case that $Y_k = X_{k+1}$. The shape of the new composite machine k is determined by

$$X_k := X_k, \; Y_k := Y_{k+1} \cup \{j\}.$$

For $k < i < m$,

$$X_i := X_{i+1} \cup \{j\}, \; Y_i := Y_{i+1} \cup \{j\}.$$

Finally,

$$X_m := \{j\}, \; Y_m := \emptyset.$$

Note that it is always the case that $X_1 = \emptyset$, $Y_i = X_{i+1}$, for $1 \leq i < m$, and $Y_m = \emptyset$.

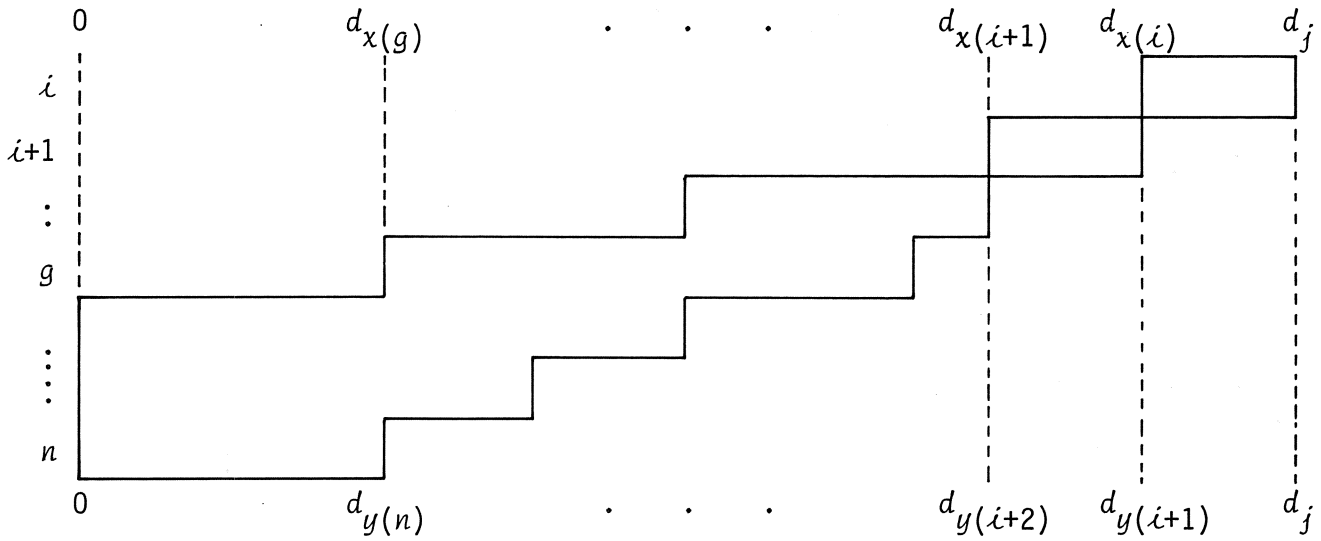


Figure 4 Composite machine i with shape X,Y.

7. DYNAMIC PROGRAMMING FORMULATION

Let us now turn the problem of minimizing the weighted number of late jobs. The values d_j are now treated as due dates, rather than absolute deadlines. We may choose to schedule job j, or not to schedule it at time d_j. (If job j is not scheduled at time d_j, it can be assumed to be processed at some later date.)

Let $S_i^{(j)}(w,X,Y)$ denote the maximum attainable processing capacity for composite machine i at time d_j, before the possible scheduling of job j, subject to the conditions that the composite machine has shape X,Y and has jobs with weights totalling exactly w charged to it. Let $\bar{S}_i^{(j)}(w,X,Y)$ be similarly defined, after job j has been considered for scheduling. If a given combination of i,j,w,X,Y is not feasible, we let $S_i^{(j)}(w,X,Y)$ or $\bar{S}_i^{(j)}(w,X,Y) = -\infty$.

Initially,

$$S_i^{(1)}(w,X,Y) = 0, \quad \text{if } w = 0,\ X = Y = \emptyset,$$

$$= -\infty, \text{ otherwise.}$$

From the principle of optimality of dynamic programming we have, for $1 \le i \le m-1$, $j = 1,2,\ldots,n$,

$$\bar{S}_i^{(j)}(w,X,Y\cup\{j\}) = \max_{\delta,\Delta}\{S_i^{(j)}(\delta,X,\Delta) + S_{i+1}^{(j)}(w-w_j-\delta,\Delta,Y)\} - p_j, \tag{7.1}$$

where it is understood that the maximization is carried out over values of δ such that $0 \le \delta \le w-w_j$ and over pairs $S_i^{(j)}(\delta,X,\Delta)$ and $S_{i+1}^{(j)}(w-w_j-\delta,\Delta,Y)$ such that

$$S_i^{(j)}(\delta,X,\Delta) \ge p_j > S_i^{(j)}(w-w_j-\delta,\Delta,Y).$$

For $j = 1,2,\ldots,n$, we also have

$$\bar{S}_m^{(j)}(w,X,Y) = S_m^{(j)}(w,X,Y), \quad \text{if } S_m^{(j)}(w-w_j,X,Y) < p_j,$$

$$= \max\{S_m^{(j)}(w,X,Y), S_m^{(j)}(w-w_j,X,Y) - p_j\}, \text{ otherwise.}$$

Also,

$$\bar{S}_i^{(j)}(w,X\cup\{j\},Y\cup\{j\}) = S_{i+1}^{(j)}(w,X,Y), \quad 1 \le i \le m-1.$$

$$\bar{S}_m^{(j)}(0,\{j\},\emptyset) = 0,$$

$$S_i^{(j+1)}(w,X,Y) = \bar{S}_i^{(j)}(w,X,Y) + \int_{d_j}^{d_{j+1}} s_i(u)du.$$

In order to justify equation (7.1) it is necessary to show that the sets of jobs charged to composite machines i and i+1 are necessarily disjoint. This can be established by showing that job h can be charged to composite machine i with shape X,Y only if either

(a) $h \in Y-X$

or

(b) $|X| = m-i+1$, $|Y| = m-i$, $\min X < h < \min Y$,

or

(c) $|X| < m-i+1$, $|Y| = m-i$, $h < \min Y$.

Since the shapes of machines i and i+1 are X,Δ and Δ,Y, respectively, this shows that a given job h cannot be charged to both of the machines i and i+1.

Now suppose that $\bar{S}_i^{(n)}(w,X,Y)$ has been computed for all feasible combinations of i,w,X and Y. Let

$$F_1(w,Y) = \bar{S}_1^{(n)}(w,\emptyset,Y),$$

$$F_{i+1}(w,Y) = \max_{\delta,\Delta}\{F_i(\delta,\Delta) + \bar{S}_{i+1}^{(n)}(w-\delta,\Delta,Y)\}.$$

The optimum value of w, i.e. the maximum total weight of jobs which can be scheduled on time, is given by the largest value of w such that $F_m(w,\emptyset)$ is finite.

The actual subset of jobs yielding an optimal schedule can be determined by recording with each $S_i^{(j)}$ or $\bar{S}_i^{(j)}$ value in the computation the set of jobs charged to the composite machine in question.

8. COMPUTATIONAL COMPLEXITY

It should be apparent that the complexity of the dynamic programming computation is determined by equation (7.1). Moreover, the computational bottleneck occurs either for $i = 1$, for which it is necessary to compute (7.1) only for $X = \emptyset$, or for $i = 2$.

Consider the case $i = 1$. We have $j = 1,2,\ldots,n$, $0 \le w \le W$, $X = \emptyset$, $|Y| \le m-2$. Hence there are $O(Wn^{m-1})$ equations for which the maximization indicated in (7.1) must be carried out. For each equation we have $0 \le \delta \le w-w_j < W$, $|\Delta| \le m-1$, or $O(Wn^{m-1})$ combinations of δ and Δ to test. Hence for $i = 1$, the computation required by equations (7.1) is bounded by time of $O(W^2n^{2m-2})$.

For $i = 2$, we have $j = 1,2,\ldots,n$, $0 \le w \le W$, $|X| \le m-1$, $|Y| \le m-3$. Hence there are $O(Wn^{2m-3})$ equations for which the maximization indicated in (7.1) must be carried out. For each equation we have $0 \le \delta \le w-w_j < W$, $|\Delta| \le m-2$, or $O(Wn^{m-2})$ combinations of δ and Δ to test. Hence for $i = 2$, the computation required by equation (7.1) is bounded by $O(W^2n^{3m-5})$.

For $m \le 3$, $2m-2 \ge 3m-5$, and for $m \ge 3$, $3m-5 \ge 2m-2$. Hence the running time is bounded by $O(W^2n^2)$ for $m = 2$, and by $O(W^2n^{3m-5})$ for $m \ge 3$. In the unweighted case, where each $w_j = 1$ and $W = n$, n^2 can be substituted for W^2 in each of the running time bounds above.

We have thus established the result that, for any fixed number of machines, the unweighted problem can be solved in polynomial time and the weighted problem in pseudopolynomial time.

9. A FULLY POLYNOMIAL APPROXIMATION SCHEME

We conclude by noting that, for any fixed number of machines, there exists a fully polynomial approximation scheme for the weighted problem. Such a scheme accepts as input the data for any given instance of the weighted problem and a real value $\varepsilon > 0$, and produces as output a feasible solution of value w, such that

$$w^* - w \leq \varepsilon w^*, \tag{9.1}$$

where w^* is the value of an optimal solution.

We propose to replace each job weight w_j by

$$v_j = \lceil w_j/K \rceil,$$

where K is a suitably chosen scale factor. We can then replace W by W/K in each of the running time bounds we have obtained.

In order for a value of K to be acceptable, i.e. in order for (9.1) to be satisfied, it is necessary that

$$Kn \leq \varepsilon LB,$$

where LB is the value of a lower bound on w^* [5].

Let

$$w_{max} = \max_j\{w_j\}.$$

Clearly

$$w_{max} \leq w^* \leq W \leq nw_{max}.$$

(We assume that any single job can be scheduled to meet its due date.) Then setting

$$K = \frac{\varepsilon}{n} w_{max},$$

we have

$$\frac{W}{K} \leq \frac{n^2}{\varepsilon}.$$

It follows that $\frac{n^2}{\varepsilon}$ can be substituted for W in each of the time bounds obtained in the previous section, and this establishes the desired result.

REFERENCES

1. T. GONZALEZ, S. SAHNI, Preemptive scheduling of uniform processor systems, *J. Assoc. Comput. Mach.* 25(1976)91-101.
2. R.L. GRAHAM, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Ann. Discrete Math.*, to appear.
3. J. LABETOULLE, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN, Preemptive scheduling of uniform machines subject to release dates, *Proc. Summer School in Combinatorial Optimization*, Urbino, Italy, 1978, to appear.
4. E.L. LAWLER, Sequencing to minimize the weighted number of tardy jobs, *Rev. Française Automat. Informat. Recherche Opérationnelle* 10.5(1976) Suppl.27-33.
5. E.L. LAWLER, Fast approximation algorithms for knapsack problems, *Math. Oper. Res.*, to appear.
6. E.L. LAWLER, J.M. MOORE, A functional equation and its application to resource allocation and sequencing problems, *Management Sci.* 16(1969) 77-84.
7. J.M. MOORE, An *n* job, one machine sequencing algorithm for minimizing the number of late jobs, *Management Sci.* 15(1968)102-109.
8. M.H. ROTHKOPF, Scheduling independent tasks with due times on a *Management Sci.* 12(1966)437-447.
9. S. SAHNI, Y. CHO, Scheduling independent tasks with due times on a uniform processor systems, *J. Assoc. Comput. Mach.*, to appear.
10. S. SAHNI, Y. CHO, Nearly on line scheduling of a uniform processor system with release times, *SIAM J. Comput.*, to appear.