stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE BW 143/81 JUNI
(DEPARTMENT OF OPERATIONS RESEARCH)

A.M.A. HARIRI & C.N. POTTS

AN ALGORITHM FOR SINGLE MACHINE SEQUENCING WITH RELEASE DATES TO MINIMISE TOTAL WEIGHTED COMPLETION TIME

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

1980 Mathematics Subject Classification: 90B35

An algorithm for single machine sequencing with release dates to minimise total weighted completion time [*)]

by

A.M.A. Hariri & C.N. Potts[**)]

ABSTRACT

Each of n jobs is to be processed without interruption on a single machine which can handle only one job at a time. Each job becomes available for processing at its release date, requires a processing time and has a positive weight. Given a processing order of the jobs, the earliest completion time for each job can be computed. The objective is to find a processing order of the jobs which minimizes the sum of weighted completion times. In this paper a branch and bound algorithm for the problem is derived. Firstly a heuristic is presented which is used in calculating the lower bound. Then the lower bound is obtained by performing a Lagrangean relaxation of the release date constraints; the Lagrange multipliers are chosen so that the sequence generated by the heuristic is an optimum solution of the relaxed problem, thus yielding a lower bound. A method to increase the lower bound by deriving improved constraints to replace the original release date constraints is given. The algorithm, which includes several dominance rules, is tested on problems with up to fifty jobs. The computational results indicate that the version of the lower bound using improved constraints is superior to the original version.

KEY WORDS & PHRASES: *single machine sequencing, release dates, total weighted completion time, branch and bound, Lagrangean relaxation, dominance, computational experience*

*) This report will be submitted for publication in a journal.

**) University of Keele, England

1. INTRODUCTION

The problem considered in this paper may be stated as follows. Each of n jobs (numbered 1,...,n) is to be processed without interruption on a single machine which can handle only one job at a time. Job i (i = 1,...,n) becomes available for processing at its release date r_i, requires a processing time p_i and has a positive weight w_i. Given a processing order σ of the jobs, the (earliest) completion time $C_i(\sigma)$ for each job i can be computed. When no ambiguity results, we abbreviate $C_i(\sigma)$ to C_i. The objective is to find a processing order of the jobs which minimizes the sum of weighted completion times $\sum w_iC_i$.

When all release dates are equal, the problem can be solved using the algorithm of SMITH [8] in which jobs are sequenced in non-increasing order of w_i/p_i. However, LENSTRA et al. [6] have shown that when jobs have arbitrary release dates and unit weights the problem is NP-hard, which indicates that the existence of a polynomial bounded algorithm is unlikely. Consequently, branch and bound algorithms have been proposed for this problem with unit weights by CHANDRA [1] and DESSOUKY & DEOGUN [3]. For the problem with arbitrary weights, RINALDI & SASSANO [7] have derived several dominance theorems. In this paper a branch and bound algorithm for the problem with arbitrary weights is derived.

In Section 2 a heuristic method for sequencing the jobs is given. A lower bound, which is computed from this sequence, is derived in Section 3 and its working is demonstrated with a numerical example. An improvement to the lower bound is presented in Section 4. Section 5 contains a statement of the branching rule and gives some dominance rules which help to reduce the size of the search tree used in the branch and bound algorithm. A complete statement of the algorithm including details of its implementation is given in Section 6. Computational experience is presented in Section 7 which is followed by some concluding remarks in Section 8.

2. THE HEURISTIC METHOD

It is well-known that computation can be reduced by using a heuristic method to find a good solution to act as an upper bound on the sum of weighted completion times prior to the application of a branch and bound algorithm. Also, in our algorithm, a sequence generated by the heuristic method is used at each node of the search tree for calculating a lower bound.

The heuristic that is used has the property that the machine will never be kept unnecessarily idle. If there is a choice of jobs for the first unfilled position in the sequence which preserves this property, one with the largest w_i/p_i is chosen. A formal statement of the method is given below.

<u>Step 1</u>. Let S be the set of all (unsequenced) jobs, let $H = 0$ and $k = 0$ and find $T = \min_{j \in S}\{r_j\}$.

<u>Step 2</u>. Find the set $S' = \{j \mid j \in S, r_j \leq T\}$ and find a job i with $i \in S'$ and with $w_i/p_i = \max_{j \in S'}\{w_j/p_j\}$.

<u>Step 3</u>. Set $k = k+1$, sequence job i in position k, set $T = T+p_i$, set $H = H+w_iT$ and set $S = S - \{i\}$.

<u>Step 4</u>. If $S = \emptyset$, then stop with the sequence generated having H as its sum of weighted completion times. Otherwise set $T = \max\{T, \min_{j \in S}\{r_j\}\}$ and go to Step 2.

We now derive sufficient conditions for the sequence generated by the heuristic to be optimum. However, some notation is introduced first. It is assumed that the jobs have been renumbered so that the sequence generated by the heuristic is $(1,\ldots,n)$ and the completion times of the jobs have been computed using $C_1 = r_1+p_1$, $C_i = \max\{r_i,C_{i-1}\} + p_i$ $(i = 2,\ldots,n)$. The jobs may be partitioned into blocks $S_1,\ldots,S_k$ as follows. Job v_j is the *last job in a block* if $C_{v_j} \leq r_i$ for $i = v_j+1,\ldots,n$. A set of jobs $S_j = \{u_j,\ldots,v_j\}$ forms a *block* if the following conditions are satisfied:

(a) $u_j = 1$ or job u_j-1 is the last job in a block;
(b) job i is not the last job in a block for $i = u_j,\ldots,v_j-1$;
(c) job v_j is the last job in a block.

Job u_j is called the *first job in a block* and, for our heuristic, has the property that $r_{u_j} \leq r_i$ for $i = u_j+1,\ldots,n$. These definitions concerning blocks were proposed by LAGEWEG et al. [5].

The sufficient conditions for the sequence generated by the heuristic to be optimum are as follows.

THEOREM 1. *The sequence* (1,...,n) *generated by the heuristic is optimum if the jobs within each block* S_j *are sequenced in non-increasing order of* w_i/p_i.

PROOF. The result is first proved for the modified problem in which the release date of each job i in S_j is set to the release date of the first job in block S_j $(j = 1,\dots,k)$. We first show that all jobs in block S_j should be sequenced before all jobs in block S_{j+1} $(j = 1,\dots,k-1)$ for this problem with reduced release dates. Consider any sequence and suppose that $i \in S_j$ is chosen so that i is as small as possible and so that job i is sequenced after a job in block $S_{j'}$ where $j < j'$. Suppose that this sequence is of the form $\sigma_1\sigma_2\sigma_3 i\sigma_4$, where σ_1 consists of all jobs in blocks $S_1,\dots,S_{j-1}$, where σ_2 consists of jobs in block S_j and where the first job of σ_3 is a job in $S_{j'}$. Consider now the new sequence $\sigma_1\sigma_2 i\sigma_3\sigma_4$. The completion time of job i in this sequence is not greater than the release date of the first job in σ_3 which is in block $S_{j'}$ since the jobs in $\sigma_2 i$ are contained in block S_j. Thus the new sequence has a smaller sum of weighted completion times. Having established that, for an optimum sequence, all jobs within a block are sequenced in adjacent positions, their ordering is determined by Smith's rule. This proves the result for the problem with reduced release dates.

We now return to the original problem obtained by increasing the release dates to their initial values. Since this increase in release dates leaves the completion times unaltered, the sequence (1,...,n) is also optimum for the original problem. □

It is seen in the next section that Theorem 1 is used in deriving our lower bound.

3. DERIVATION OF THE LOWER BOUND

The method used to obtain a lower bound is similar to the multiplier adjustment method proposed by VAN WASSENHOVE [9] for minimizing $\sum w_i C_i$ when jobs have zero release date and have deadlines. We obtain a lower bound by performing a Lagrangean relaxation of each release date constraint $C_i \geq r_i+p_i$ $(i = 1,\dots,n)$ after which it is replaced by a weaker constraint $C_i \geq r_i^*+p_i$ for some $r_i^* \leq r_i$. This yields the Lagrangean problem

$$(1) \qquad L(\lambda) = \min\{\textstyle\sum_{i=1}^n w_iC_i + \sum_{i=1}^n \lambda_i(r_i + p_i - C_i)\},$$

where $\lambda = (\lambda_1,\dots,\lambda_n)$ is a vector of non-negative multipliers; the minimization is over all processing orders of the jobs with C_i $(i = 1,\dots,n)$ subject to machine capacity constraints and to the constraints $C_i \geq r_i^*+p_i$. We can write (1) as

$$L(\lambda) = \min\{\textstyle\sum_{i=1}^n w_i^*C_i\} + \sum_{i=1}^n \lambda_i(r_i+p_i),$$

where $w_i^* = w_i-\lambda_i$ $(i = 1,\dots,n)$. Thus, the Lagrangean problem is of the same form as the original problem but each job i has a new release date r_i^* and a new weight w_i^*. The choice of new release dates and of multipliers is discussed next. However, we shall restrict our choice of multipliers to the range $0 \leq \lambda_i \leq w_i$ $(i = 1,\dots,n)$ to ensure that $L(\lambda)$ does not become arbitrarily small. One possible approach is to set $r_i^* = 0$ so that the Lagrangean problem can be solved using Smith's rule. The value of λ which maximizes $L(\lambda)$ can then be found using the subgradient optimization method. However, this might entail much computation without guarantee of a tight lower bound. We prefer to retain the original values of the release dates, i.e. to set $r_i^* = r_i$ $(i = 1,\dots,n)$, but restrict the choice of multipliers so that the Lagrangean problem can be solved easily. This can be achieved by maximizing $L(\lambda)$ subject to the condition that the sequence generated by the heuristic solves the Lagrangean problem by yielding weights w_i^* $(i = 1,\dots,n)$ which satisfy the conditions of Theorem 1. Thus we require for each block S_j that

$$(w_i-\lambda_i)/p_i \leq (w_{i-1} - \lambda_{i-1})/p_{i-1} \qquad \text{for } i = u_j+1,\dots,v_j.$$

It is clear that $L(\lambda)$ is maximized by choosing

$$(2) \qquad \lambda_i = \left\{ \begin{array}{ll} 0 & \text{if } i = u_j, \\ \max\{0, w_i + (\lambda_{i-1} - w_{i-1}) p_i / p_{i-1}\} & \text{if } i = u_j + 1, \ldots, v_j \end{array} \right\} (j = 1, \ldots, k).$$

Having found C_i $(i = 1,\ldots,n)$ using the sequence generated by the heuristic and λ_i $(i = 1,\ldots,n)$ using (2), our lower bound can be written as

$$(3) \qquad LB = \textstyle\sum_{i=1}^{n} w_i C_i + \sum_{i=1}^{n} \lambda_i (r_i + p_i - C_i).$$

EXAMPLE. The data for the example is summarized in the first three rows of Table 1. The jobs have already been renumbered so that the sequence generated by the heuristic method is $(1,\ldots,10)$.

Table 1

Data for the example.

i	1	2	3	4	5	6	7	8	9	10
r_i	0	6	9	15	21	22	23	25	22	22
p_i	5	4	4	3	6	2	10	5	8	9
w_i	10	3	8	8	3	6	10	4	6	6
C_i	5	10	14	18	27	29	39	44	52	61
λ_i	0	0	5	0	0	5	5	3/2	2	3/2
$\lambda_i(C_i-r_i-p_i)$	0	0	5	0	0	25	30	21	44	45

Having applied the heuristic method, the completion times of the jobs are computed. These are shown the row 4 of Table 1. The sum of weighted completion times is 1835. The blocks obtained from this sequence are $S_1 = \{1\}$, $S_2 = \{2,3\}$, $S_3 = \{4\}$ and $S_4 = \{5,6,7,8,9,10\}$. The multipliers, obtained from (2) are shown in row 5 of Table 1. The value of the lower bound is computed from (3) using the bottom row of Table 1. This gives $LB = 1835 - 170 = 1665$.

4. THE IMPROVED LOWER BOUND

We assume that the multipliers defined in the previous section have been computed using (2). Suppose that the jobs are ordered within each block in non-decreasing order of multipliers to give a permutation $\pi = (\pi(1),\ldots,\pi(n))$ with the property that $S_j = \{\pi(u_j),\ldots,\pi(v_j)\}$ and that $\lambda_{\pi(u_j)} \le \ldots \le \lambda_{\pi(v_j)}$ $(j = 1,\ldots,k)$. It is clear from (2) that $\lambda_{\pi(u_j)} = 0$ since the first job in a block always yields a zero multiplier. We now define

$$S_j^{(h)} = S_j^{(h-1)} - \{\pi(u_j+h-1)\} \quad (h=1,\ldots,v_j-u_j,\ j=1,\ldots,k),$$

where $S_j^{(0)} = S_j$ and

$$\mu_j^{(h)} = \lambda_{\pi(u_j+h)} - \lambda_{\pi(u_j+h-1)} \quad (h=1,\ldots,v_j-u_j,\ j=1,\ldots,k).$$

The set $S_j^{(h)}$ is obtained from the set $S_j^{(h-1)}$ by deleting a job having the smallest multiplier and $\mu_j^{(h)}$ is the difference in value between the multiplier of the job deleted and the smallest multiplier of the remaining jobs. From these definitions, we can rewrite (3) as

$$(4) \qquad LB = \sum_{i=1}^{n} w_i C_i + \sum_{j=1}^{k} \sum_{h=1}^{v_j-u_j} \mu_j^{(h)} (b_j^{(h)} - \sum_{i\in S_j^{(h)}} C_i),$$

where $b_j^{(h)} = \sum_{i\in S_j^{(h)}} (r_i+p_i)$ $(h = 1,\ldots,v_j-u_j,\ j = 1,\ldots,k)$. (It is assumed that any summation is zero when its lower limit exceeds its upper limit.) Clearly $b_j^{(h)}$ is a lower bound on $\sum_{i\in S_j^{(h)}} C_i$. However, if a better lower bound can be found, it is possible to increase LB. To obtain the best possible bound on the sum of completion times of jobs having release dates would require the solution of an NP-hard problem [6]. Since this is computationally expensive, we prefer to obtain a lower bound on $\sum_{i\in S_j^{(h)}} C_i$ by solving the corresponding preemptive scheduling problem in which the processing of any job can be interrupted and resumed at a later time. The preemptive problem is solved by the following algorithm of CONWAY et al. [2]. At any time when a job is completed or when a new job becomes avaliable for processing, the job which is processed next is one with the shortest remaining processing time. If $\beta_j^{(h)}$ denotes the sum of completion times for the jobs in $S_j^{(h)}$ when they are sequenced using this shortest remaining processing time rule, we have the following improved lower bound:

$$LB' = LB + \sum_{j=1}^{k} \sum_{h=1}^{v_j-u_j} \mu_j^{(h)}(\beta_j^{(h)} - b_j^{(h)}).$$

Since $\beta_j^{(h)} \geq b_j^{(h)}$, it is clear that $LB' \geq LB$.

EXAMPLE. We consider again the example given in the previous section. We have $\pi = (1,2,3,4,5,8,10,9,6,7)$ with the ordering between job 8 and job 10 and between job 5 and job 7 being arbitrary due to equality of multipliers. We now compute

$S_1^{(0)} = \{1\}$

$S_2^{(0)} = \{2,3\}$, $S_2^{(1)} = \{3\}$ $\mu_2^{(1)} = 5$

$S_3^{(0)} = \{4\}$

$S_4^{(0)} = \{5,8,10,9,6,7\}$, $S_4^{(1)} = \{8,10,9,6,7\}$ $\mu_4^{(1)} = 1.5$

$S_4^{(2)} = \{10,9,6,7\}$ $\mu_4^{(2)} = 0$, $S_4^{(3)} = \{9,6,7\}$ $\mu_4^{(3)} = 0.5$

$S_4^{(4)} = \{6,7\}$ $\mu_4^{(4)} = 3$, $S_4^{(5)} = \{7\}$ $\mu_4^{(5)} = 0$.

Clearly, we have $b_2^{(1)} = \beta_2^{(1)} = 13$. Solving the preemptive scheduling problem for jobs in $S_4^{(1)}$ yields $\beta_4^{(1)} = 193$ compared with $b_4^{(1)} = 148$. Similarly, $\beta_4^{(2)} = 148$ with $b_4^{(2)} = 118$, $\beta_4^{(3)} = 98$ with $b_4^{(3)} = 87$, $\beta_4^{(4)} = 58$ with $b_4^{(4)} = 57$, $\beta_4^{(5)} = 33$ with $b_4^{(5)} = 33$. (For this particular problem $\beta_4^{(2)}$ and $\beta_4^{(5)}$ need not have been computed since $\mu_4^{(2)} = \mu_4^{(5)} = 0$.) Thus $LB' = LB + 76 = 1741$.

5. DOMINANCE RULES

If it can be shown that an optimum solution can always be generated without branching from a particular node of the search tree, then that node is dominated and can be eliminated. Dominance rules usually specify whether a node can be elminated before its lower bound is calculated. Clearly, dominance rules are particularly useful when a node can be eliminated which has a lower bound that is less than the optimum solution.

In our search tree, nodes at level ℓ represent initial partial sequences in which jobs in the first ℓ positions have been fixed. The merits of this branching rule are discussed in the next section. The following results will show when any of the immediate successors of the node corresponding to an initial partial sequence σ are dominated. We assume that $\sigma = \sigma_1 h$, whenever σ is not empty. Also we define $\bar{\sigma}$ to be the set of jobs not sequenced in σ and we define the earliest start time of these unsequenced jobs as $T = \max\{C_h(\sigma), \min_{i \in \bar{\sigma}}\{r_i\}\}$. (The obvious generalized notation that $C_h(\sigma)$ denotes the (earliest) completion time of job h in the partial sequence σ is adopted.)

The first of our dominance theorems is a result of RINALDI & SASSANO [6] . For completeness the proof is outlined.

THEOREM 2 (RINALDI & SASSANO). *If job* i *is chosen with* $i \in \bar{\sigma}$ *and with* $w_i/p_i = \max_{j \in \bar{\sigma}}\{w_j/p_j\}$ *and if* $\max\{r_i,T\} \leq \max\{r_j,T\}$ *for any* $j \in \bar{\sigma}$, *where* $j \neq i$, *then* σj *is dominated.*

PROOF. Consider any sequence $\sigma j \sigma' i \sigma''$ having σj as initial partial sequence. Job i can be interchanged with the job sequenced immediately before it without increasing the sum of weighted completion times. After the repeated application of this process, the sequence $\sigma i j \sigma' \sigma''$ will result which does not have σj as an initial partial sequence. □

If in Theorem 2 we have $r_i \leq T$, then the node corresponding to σ will have only one immediate successor σi. The lower bound for this successor is identical with that of its parent node and need not be computed again.

The next result is due to DESSOUKY & DEOGUN [3]. It states that the machine should not be kept idle throughout a time interval within which another job can be completely processed. Again, the proof is outlined.

THEOREM 3 (DESSOUKY & DEOGUN). *If* $r_j \geq C_i(\sigma i)$ *for any* $i,j \in \bar{\sigma}$, *then* σj *is dominated.*

PROOF. Given any sequence $\sigma j \sigma' i \sigma''$ having σj as an initial partial sequence, a new sequence $\sigma i j \sigma' \sigma''$ can be formed in which job i has a smaller completion time and in which the jobs in σ' and σ'' do not have a larger completion time. This new sequence does not have σj as an initial partial sequence. □

It is apparent that the conditions of Theorem 3 are most likely to be satisfied when job i is chosen with $C_i(\sigma i)$ as small as possible. It is expected that Theorem 3 will be most effective at reducing the size of the search tree when release dates have a large range.

Our final result is a consequence of dynamic programming. If the final two jobs of a partial sequence can be interchanged without increasing the sum of weighted completion times of jobs in the partial sequence and without increasing the time at which the machine becomes available to process the next unsequenced job, then this partial sequence is dominated. The importance of this type of dominance rule is often overlooked in single machine sequencing. Recalling that $\sigma = \sigma_1 h$, our dominance theorem is as follows.

THEOREM 4. *If* $C_h(\sigma_1 jh) \le C_j(\sigma_1 hj)$ *and if* $w_j C_j(\sigma_1 jh) + w_h C_h(\sigma_1 jh) \le w_h C_h(\sigma_1 hj) + w_j C_j(\sigma_1 hj)$ *for any* $j \in \bar{\sigma}$, *then* $\sigma_1 hj$ *is dominated.*

Care must be taken when both of the conditions of Theorem 4 hold with equality that only one of the partial sequences $\sigma_1 hj$ and $\sigma_1 jh$ is discarded. It is possible to derive other dynamic programming dominance conditions involving the interchange of another pair of jobs or involving a larger group of jobs, but they are unlikely to be very effective once the three other theorems have been applied.

6. THE ALGORITHM

The branching rule is discussed first. As was stated in the previous section, a node at level ℓ of the search tree corresponds to an initial partial sequence in which jobs in the first ℓ positions are fixed. This procedure has the advantage that once a job has been sequenced, its completion time is immediately computed and it can be discarded from consideration in all successor nodes. Alternatively, if nodes correspond to final partial sequences, completion times of sequenced jobs depend on the processing order of unsequenced jobs. Before any new node is created, the dominance rules of the previous section are checked. If job i can be found satisfying the conditions of Theorem 2 with $r_i \le T$, then a single successor node is created whose lower bound is the same as that of its parent. In other cases as many nodes as possible are eliminated using Theorem 2. Then a job i is found with $C_i(\sigma i)$ as small as possible and the remaining nodes are checked for dominance using Theorem 3. Theorem 4 is applied to all nodes which have not been eliminated.

For each node of the search tree which cannot be eliminated by dominance rules, a lower bound is calculated. Firstly, the release date of each unsequenced job i is adjusted by setting $r_i = \max(r_i, T)$, where T denotes the earliest start time of unsequenced jobs. Then the heuristic method described in Section 2 and the lower bounding methods described in Section 3 and Section 4 are applied to the unsequenced jobs and the contributions of sequenced jobs are added. At level ℓ of the search tree where there are $\bar{\ell} = n-\ell$ unsequenced jobs, the heuristic requires $O(\bar{\ell} \log \bar{\ell})$ steps. A further $\bar{\ell}$ steps are required to compute LB. If LB exceeds the value of a solution already computed, then this node is discarded. Otherwise, the lower bound LB' is computed. Since the solution of a preemptive scheduling problem with $\bar{\ell}$ jobs requires $O(\bar{\ell} \log \bar{\ell})$ steps, a further $O(\bar{\ell}^2 \log \bar{\ell})$ steps are required to solve the $O(\bar{\ell})$ preemptive scheduling problems. To summarise, LB requires $O(\bar{\ell} \log \bar{\ell})$ steps and LB' requires $O(\bar{\ell}^2 \log \bar{\ell})$ steps.

Finally, our search strategy is given. A newest active node search is used which selects a node from which to branch which has the smallest lower bound amongst nodes in the most recently created subset.

7. COMPUTATIONAL EXPERIENCE

The algorithm described in the previous section which uses both LB and LB' and a similar algorithm which uses only LB were tested on problems with 20, 30, 40 and 50 jobs. For each job i, an integer processing time p_i from the uniform distribution [1,100] and an integer weight w_i from the uniform distribution [1,10] were generated. Since the range of release dates is likely to influence the effectiveness of the algorithms, an integer release date for each job i was generated from the uniform distribution [0,50.5nR], where R controls the range of the distribution. The value 50.5n measures the expected total processing time. For each selected value of n, five problems were generated for each of the R values 0.2, 0.4, 0.6, 0.8, 1.0, 1.25, 1.5, 1.75, 2.0 and 3.0 producing fifty problems for each value of n.

The algorithms were coded in FORTRAN IV and run on a CDC 7600 computer. Average computation times and average numbers of nodes are given in Table 2. Whenever a problem was not solved within the time limit of 60 seconds, computation was abandoned for that problem. Thus in some cases the figures given in Table 2 will be lower bounds on the average computation times and the average numbers of nodes. Numbers of unsolved problems for the different values of R are listed in Table 3.

Table 2

Average computation times and average numbers of nodes

n	Lower bound LB		Lower bound LB'	
	Average computation time[a]	Average number of nodes	Average computation time[a]	Average number of nodes
20	0.08	351	0.06	170
30	3.23[b]	10439[b]	1.47	2203
40	17.30[b]	40991[b]	14.89[b]	24651[b]
50	33.09[b]	65883[b]	30.58[b]	41255[b]

[a] Times are in CPU seconds

[b] Lower bounds because of unsolved problems

Table 3

Numbers of unsolved problems

	n	R									
		0.2	0.4	0.6	0.8	1.0	1.25	1.5	1.75	2.0	3.0
LB	20	0	0	0	0	0	0	0	0	0	0
	30	0	0	1	0	0	0	0	0	0	0
	40	0	0	3	4	1	2	0	0	0	0
	50	0	4	5	5	5	3	1	0	0	0
LB'	20	0	0	0	0	0	0	0	0	0	0
	30	0	0	0	0	0	0	0	0	0	0
	40	0	0	2	3	1	1	0	0	0	0
	50	0	3	5	5	5	2	1	0	0	0

It is clear from the average computation times that LB' is superior to LB. The difference in performance is most apparent for the thirty job problems.

For n = 40 and n = 50 the true difference between LB and LB' in Table 2 is disguised by the unsolved problems. It can also be seen from Table 2 that the average computation time per node is considerably less for LB as is expected.

Table 3 shows that there are a total of 34 unsolved problems for LB compared with a total of 28 for LB' which again demonstrates the superiority of LB'. For both bounds, the problems with small R and large R are easiest. This is expected because for small R the release dates become unimportant once a few jobs have been sequenced enabling Theorem 2 to restrict the numbers of immediate successor nodes to one. However, when R is large the release dates become more important than the processing times and weights allowing Theorem 3 to successfully limit the size of the search tree. The hardest problems occur when R = 0.6, R = 0.8 and R = 1.0.

8. CONCLUDING REMARKS

The algorithm using the lower bound LB' is satisfactory for solving small and medium sized problems. However, a sharper lower bound is needed to cut down the size of the search tree when the number of jobs exceeds thirty.

One way in which the algorithm might be improved is to use the partitioning idea proposed by RINALDI & SASSANO [7]. This states that if an optimum sequence σ of a subset of the original jobs can be found such that the release dates of all jobs not sequenced in σ are not less than the completion times of jobs in σ, then an optimum sequence to the complete problem exists which has σ as an initial partial sequence. When such a subsequence σ can be found, the remaining problem involving all jobs not sequenced in σ can be solved independently. However, the best way to find the necessary subset of jobs requires investigation.

The lower bounds LB and LB' are also valid lower bounds for the preemptive version of our problem. They could, with a suitable branching rule and with dominance rules, be used in a branch and bound algorithm for this preemptive scheduling problem which is NP-hard [4]. Our bounds can also be applied to the possibly more realistic non-preemptive problem in which unforced machine idle time is not allowed.

As well of being of interest in its own right, the solution of the problem considered in this paper might prove useful in obtaining lower bounds for flow-shop and job-shop problems based on Lagrangean relaxation. This seems to be worthy of future research.

ACKNOWLEDGEMENTS

The authors are grateful to J.K. Lenstra for his comments on a preliminary version of this paper. The second author is grateful to the Mathematisch Centrum, Amsterdam, for helping to finance a visit to the Mathematisch Centrum where this research was completed.

REFERENCES

[1] CHANDRA, R., *On $n/1/\bar{F}$ dynamic deterministic systems*, Naval Res. Logist. Quart. 26 (1979) 537-544.

[2] CONWAY, R.W., MAXWELL W.L. & L.W. MILLER, *Theory of scheduling*, Addison-Wesley, Reading, MA (1967).

[3] DESSOUKY, M.I. & J.S. DEOGUN, *Sequencing jobs with unequal ready times to minimize mean flow time*, SIAM J. Comput. 10 (1981) 192-202.

[4] LABETOULLE, J., LAWLER, E.L., LENSTRA, J.K. & A.H.G. RINNOOY KAN, *Preemptive scheduling of uniform processors subject to release dates*, Report BW 99, Mathematisch Centrum, Amsterdam (1979).

[5] LAGEWEG, B.J., LENSTRA, J.K. & A.H.G. RINNOOY KAN, *Minimizing maximum lateness on one machine: computational experience and some applications*, Statistica Neerlandica 30 (1976) 25-41.

[6] LENSTRA, J.K., RINNOOY KAN, A.H.G. & P. BRUCKER, *Complexity of machine scheduling problems*, Ann. Discrete Math. 1 (1977) 343-362.

[7] RINALDI, G. & A. SASSANO, *On a job scheduling problem with different ready time: some properties and a new algorithm to determine the optimal solution*, Report R 7724, Istit. di Automatica, Universita di Roma (1977).

[8] SMITH, W.E., *Various optimizers for single-stage production*, Naval Res. Logist. Quart. 3 (1956) 59-66.

[9] VAN WASSENHOVE, L., *Special-purpose algorithms for one machine sequencing problems with single and composite objectives*, Ph.D. Thesis, Industrieel Beleid, Katholieke Universiteit Leuven (1979).