JOB SHOP SCHEDULING WITH UNIT TIME OPERATIONS UNDER RESOURCE CONSTRAINTS AND RELEASE DATES

Jayme Luiz Szwarcfiter

NCE 0185

September 1985

Núcleo de Computação Eletrônica
Caixa Postal 2324
20001 - Rio de Janeiro - RJ
Brasil

JOB SHOP SCHEDULING WITH UNIT TIME OPERATIONS UNDER RESOURCE CONSTRAINTS AND RELEASE DATES

Jayme Luiz Szwarcfiter
Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica

ABSTRACT

We consider the problem of job shop scheduling with m machines and n jobs J_i, each consisting of ℓ_i unit time operations. There are s distinct resources R_h and a quantity q_h available of each one. The execution of the j-th operation of J_i requires the presence of u_{ijh} units of R_h, $1 \leq i \leq n$, $1 \leq j \leq \ell_i$ and $1 \leq h \leq s$. In addition, each J_i has a release date r_i, that is J_i cannot start before time r_i. We describe algorithms for finding schedules having minimum length or sum of completion times of the jobs. Let $\ell = \max\{\ell_i\}$ and $u = |\{u_{ijh}\}|$. If m., s, u and ℓ are fixed then both algorithms terminate within polynomial time.

RESUMO

Considera-se o problema de job shop scheduling com m máquinas e n jobs J_i, cada qual consistindo de ℓ_i operações de tempo unitário. Existem s recursos distintos R_h, cada um disponível em quantidade q_h. A execução da j-ésima operação de J_i requer a presença de u_{ijh} unidades de R_h, $1 \leq i \leq n$, $1 \leq j \leq \ell_i$ e $1 \leq h \leq s$. Em adição, cada J_i possui uma data de liberação r_i, isto é, J_i não pode ser iniciado antes de r_i. São descritos algoritmos para determinar schedules possuindo comprimento ou tempo médio de término mínimos. Seja $\ell = \max\{\ell_i\}$ e $u = |\{u_{ijh}\}|$. Se m, s, u e ℓ são fixos então ambos os algoritmos terminam em tempo polinomial.

1. INTRODUCTION

We consider the job shop problem π_0 with m machines and n Jobs J_i, $1 \leq i \leq n$, each with ℓ_i unit time operations. There are s resources available in arbitrary quantities q_h and the k-th operation of J_i requires an arbitrary number u_{ikh} of units of the h-th resource, $1 \leq i \leq n$, $1 \leq k \leq \ell_i$ and $1 \leq h \leq s$. In addition, each job has an integer release date, which applies to its first operation. We describe dynamic programming algorithms which construct schedules for π_0 having minimum length C_{max} or sum of completion times ΣC, respectively. Let $\ell = \max\{\ell_i\}$, $q = \max\{q_h\}$ and $u = |\{u_{ihk}\}|$. If m, s, ℓ and u are fixed then both algorithms terminate within polynomial time

If ℓ is arbitrary then the minimization problems become NP-hard, even if m = 3 and with no resources [4], but the C_{max} minimization for the corresponding two machine case can be solved in polynomial time [3]. However, it turns again NP-hard when one unit of one resource is added [2].

If u is arbitrary NP-hardness occurs even if π_0 is restricted to a flow shop problem with m = 2 and s = 2 or 1, respectively according to the minimization criteria C_{max} or ΣC [5].

If s is arbitrary π_0 becomes NP-hard already when restricted to a flow shop problem with m = 2 and q = u = 1 [1]

Finally, when m is arbitrary it might be worth mentioning NP-hardness of one of its restrictions, namely a flow shop problem having s = q = u = 1 [1].

2. A BASIC ALGORITHM

Denote by π_0 the job shop scheduling problem having jobs $J(\pi_0) = \{J_1,\ldots,J_n\}$ and machines $\{M_1,\ldots,M_m\}$. Each J_i consists of a sequence of unit time operations $0(i,1),\ldots,0(i,\ell_i)$, and if $k<\ell_i$ then $0(i,k)$ must preceed $0(i,k+1)$. A ℓ_i-tuple $P_i = (p_{i1},\ldots, p_{i\ell_i})$ specifies the machine assignment of J_i, i.e. $0(i,k)$ is to be executed in machine $M_{p_{ik}}$, $1\leqslant k\leqslant \ell_i$. $0(i,1)$ and $0(i,\ell_i)$ are respectively the head and tail of J_i.

In addition, π_0 has resources $\{R_1,\ldots,R_s\}$, with a quantity $q_h \geqslant 0$ available of each R_h, $1\leqslant h\leqslant s$. A $s.\ell_i$-tuple $U_i = (U_{i11},\ldots,U_{ikh},\ldots,U_{i\ell_i s})$ describes the resource requirements of J_i, $1\leqslant k\leqslant \ell_i$ and $1\leqslant h\leqslant s$. That is, the execution of $0(i,k)$ requires the presence of $u_{ikh} \geqslant 0$ units of each R_h. The resource constraints imply that no subset of simultaneous operations in a feasible schedule can require together more than q_h units of R_h, $1\leqslant h\leqslant s$. We suppose $u_{ikh} \leqslant q_h$ in all cases, otherwise there is no feasible schedule. Let U be the set of all distinct u_{ikh} and $u = |U|$ Finally, each J_i has a release date $r_i \geqslant 0$, that is the operations of J_i can not start execution before time r_i.

The completion time of $J_i \in J(\pi_0)$ is that of its tail, relative to time zero. Below are described algorithms for respectively minimizing the length C_{max} or the sum of completion times ΣC, of a feasible schedule for π_0. The parameters ℓ, u, m and s are supposed to be fixed.

We employ dynamic programming, succesively decomposing a subproblem π into new ones of smaller size. An integer initial time t is additionally associated to each π, meaning that any feasible schedule for π must start at time t. Denote by $J(\pi)$ and

$O(\pi)$ respectively the sets of jobs and operations of π. In all cases, $t \geq r_{min}(\pi)$, where $r_{min}(\pi)$ equals zero whenever $J(\pi)=\Phi$ and $\min \{r_i | J_i \in J(\pi)\}$ otherwise.

A <u>feasible start of</u> π is a non-empty subset of heads $H \subset O(\pi)$ satisfying,

$$O(i,k) \in H \Rightarrow k=1 \text{ and } r_i \leq t \quad \ldots\ldots\ldots\ldots \quad (1)$$

$$O(i,1),\ O(j,1) \in H \Rightarrow M_{p_{i1}} \neq M_{p_{j1}} \quad \ldots\ldots\ldots\ldots\ldots \quad (2)$$

and

$$\sum_{O(i,1) \in H(\pi)} u_{i1h} \leq q_h, \text{ for each } 1 \leq h \leq s \quad \ldots\ldots\ldots \quad (3)$$

That is, H is a subset of operations which can be scheduled simultaneously to start a feasible schedule for π. Denote by H^* the set of all possible feasible starts of π.

Let $\pi(H)$ be the subproblem which satisfies $O(\pi(H))=O(\pi)-H$, that is $\pi(H)$ is the subproblem obtained by removing from π the feasible start H. Call $\pi(H)$ an <u>immediate successor</u> of π. Let $\pi_1, \ldots, \pi_k$, $k \geq 1$, be a sequence of subproblems such that π_{i+1} is an immediate successor of π_i, $1 \leq i < k$. Then π_k is a <u>successor</u> of π_1. Denote by π^* and π_I^* the sets of all successors and immediate successors of π, respectively.

Denote by $\lambda(\pi,t)$ and $\tau(\pi,t)$ respectively the minimum values of C_{max} and ΣC, for the subproblem π with initial time t. The following lemma describes the decomposition employed.

<u>Lemma 1</u>:

$$\lambda(\pi,t) = \begin{cases} 0, \text{ if } J(\pi)=\Phi. \text{ Otherwise} \\ \min_{H \in H^*} \{\max\{t+1,\ \lambda(\pi(H),\ t_H)\}\} \end{cases} \quad \ldots (4)$$

and

$$\tau(\pi,t) = \begin{cases} 0, \text{ if } J(\pi)=\phi. \text{ Otherwise} \\ \min_{H\in H^*}\{(t+1).|T(H)| + \tau(\pi(H), t_H)\} \end{cases} \quad \ldots (5)$$

where $t_H = \{\max t+1, r_{min}(\pi(H))\}$ and $T(H)$ is the subset of tails of π belonging to H.

Proof: The case of λ is trivial, see figure 1. For τ, the contribution to $\tau(\pi(H), t_H)$ of the completion time of any job of $\pi(H)$ is the same as that to $\tau(\pi,t)$. In addition, each operation $O(i,1) \in H$ contributes to $\tau(\pi,t)$ with t+1 units if it is a tail of π, and zero units otherwise.

The basic algorithm can now be formulated. Given π_0, let $t_0 = r_{min}(\pi_0)$. Then compute $\lambda(\pi_0,t_0)$ or $\tau(\pi_0,t_0)$ using (4) or (5) respectively. Each feasible start $H\in H^*$ can be obtained by applying (1)-(3).

The number of subproblems considered by the above algorithm is exponential in n. The following section describes a method which restricts the search to a polynomial number of them.

3. A POLYNOMIAL TIME ALGORITHM

We now describe a more efficient method for computing recurrences (4)-(5).

Let X be the set of f_j-tuples $X_j = (x_{j1},\ldots,x_{jf_j})$, for all $1\le f_j\le \ell$, such that $x_{jk} \in Z_m^+$, $1\le j\le |X|$ and $1\le k\le f_j$. Denote by Y the set of $s.g_j$-tuples $Y_j = (Y_{j11},\ldots,Y_{jg_js})$, for all $1\le g_j\le \ell$ satisfying $Y_{jkh} \in U$, $1\le j\le |Y|$, $1\le k\le g_j$ and $1\le h\le s$. In other words, X and Y are the sets of all possible machine assignments and resource requirements, respectively.

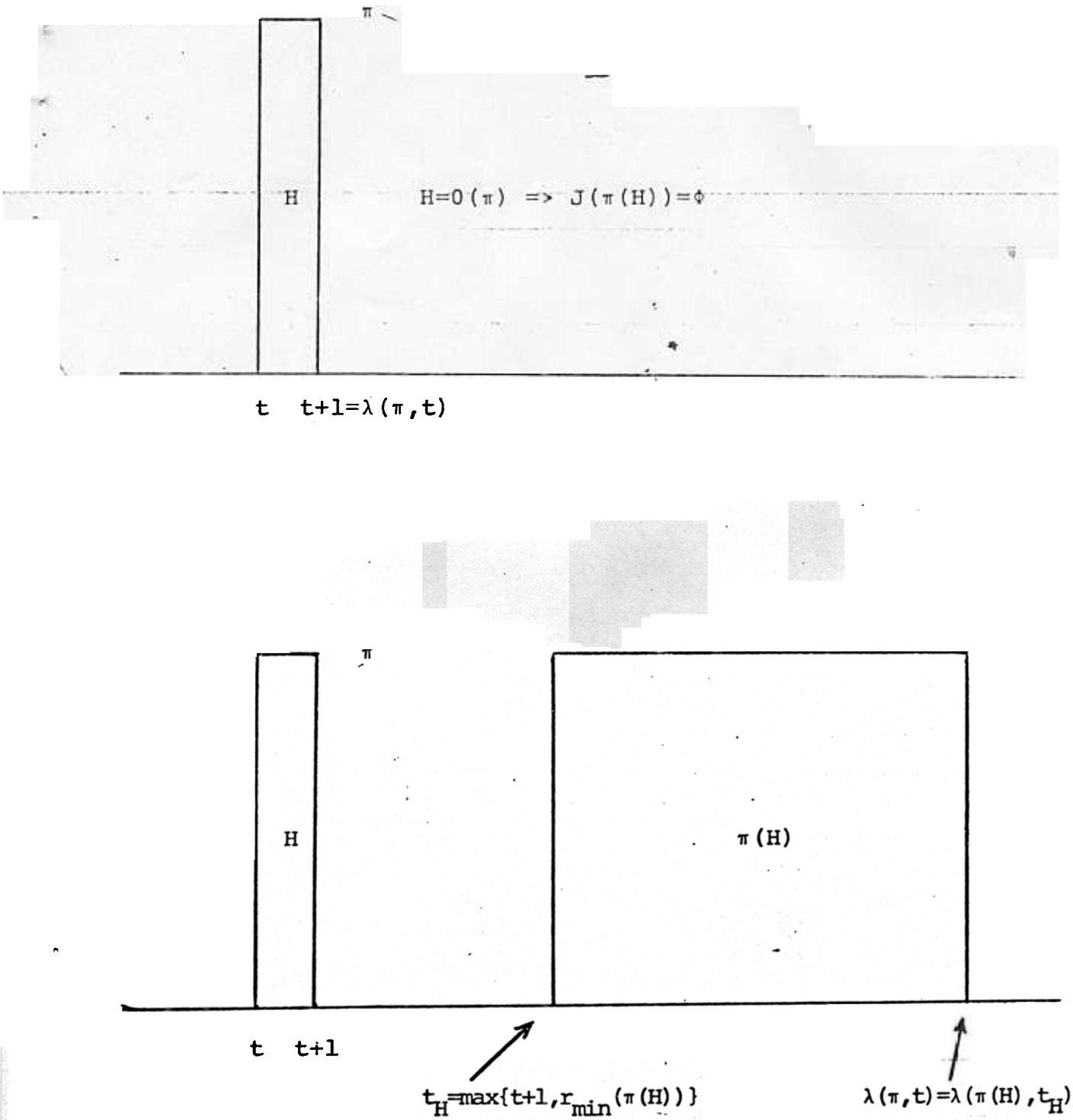

FIGURE 1: C_{max} Minimization

Lemma 2: $|X| = m(m^{\ell}-1)/(m-1$ and

$$|Y| = (u+1)^{s}\,[(u+1)^{s\ell}-1]/[(u+1)^{s}-1]$$

The proof follows from a simple counting.

The profile of π is a matrix $S(\pi)$ with elements,

$$s(i,j) = |\{J_k \in J(\pi) \quad P_k = X_i \text{ and } U_k = Y_j\}|, \quad 1\le i\le|X| \text{ and } 1\le j\le|Y| \qquad (6)$$

Let t be the initial time of π. A t-profile $S_t(\pi)$ is a profile of π restricted to the jobs with release date $\le$ t, i.e. a matrix with elements,

$$s_t(i,j) = |\{J_k \in J(\pi) \mid P_k = X_i,\ U_k = Y_j \text{ and } r_k \le t\}| \quad 1\le i\le|X| \text{ and } 1\le j\le|Y| \qquad (7)$$

The use of profiles in the solution of the minimization problems is given by the lemma below.

Lemma 3: Let π', π'' be subproblems with profiles S', S", initial times t', t" respectively, and both obtained from zero or more applications of (4) or (5). Then,

S' = S" and t' = t" =>

$\lambda(\pi',t') = \lambda(\pi'',t'')$ and $\tau(\pi',t') = \tau(\pi'',t'')$

Proof: Select if possible, $J_i \in J(\pi')$ such that $r_i \ge t'$. Since π'' has been derived from (4) or (5), its generation followed a sequence of subproblems $\pi_1,\ldots,\pi_k$, where $\pi_1 = \pi_0$, $\pi_k = \pi''$ and π_{j+1} is an immediate successor of π_j, $1\le j<k$. Since the initial times of the subproblems increase with k we concluse that $J_i \in J(\pi_j)$, $1\le j\le k$, that is $J_i \in J(\pi'')$. Repeat the above argument, until all unselected jobs $J_h \in J(\pi')$ have release dates < t'. But in this case, we can increase r_h to t' with no influence in C_{max} or

ΣC. Hence S'=S" and t'=t" imply that π' and π'' can be transformed into identical subproblems.

A subset $\pi_I^c \subset \pi_I^*$ is an immediate cover for π when,

(i) $\pi', \pi'' \in \pi_I^c$ and $\pi' \neq \pi'' \Rightarrow S(\pi') \neq S(\pi'')$

and

(ii) $\pi' \in \pi_I^* \Rightarrow \exists\ \pi'' \in \pi_I^c$ such that $S(\pi') = S(\pi'')$,

that is, π_I^c is a subset of subproblems of π having distinct profiles and covering the possible profiles among all immediate successors of π.

Similary, a subset $\pi^c \subset \pi^*$ is a cover for π when it contains exactly one subproblem of π representing each distinct profile among all successors of π.

A feasible start cover $H^c \subset H^*$ is a subset of feasible starts which spans an immediate cover for π, that is $H \in H^c$ iff $\pi(H) \in \pi_I^c$.

If follows from lemma 3 that in order to compute $\lambda(\pi,t)$ or $\tau(\pi,t)$ using (4) or (5) respectively, it is sufficient to consider the minimizations of the recurrences with the variation of H within a feasible start cover H^c of π. We now describe a method for computing H^c.

Let $V_t(\pi) \subset Z^m_{|X|} \times Z^m_{|Y|}$ be the set of pairs

$v_t = ((i_1,\ldots,i_m), (j_1,\ldots,j_m)) \neq ((0,\ldots,0), (0,\ldots,0))$ such that

$$i_k \neq 0 \text{ iff } j_k \neq 0,\ 1 \leq k \leq m \qquad \ldots (8)$$

$$i_k \neq 0 \Rightarrow x_{i_k 1} = k \text{ and } s_t(i_k, j_k) \neq 0,\ 1 \leq k \leq m \qquad \ldots (9)$$

and

$$\sum_{k \in \alpha(v_t)} Y_{j_k 1h} \leq q_h, \quad 1 \leq h \leq s, \qquad \ldots (10$$

where $\alpha(v_t) = \{k \mid i_k \neq 0, \ 1 \leq k \leq m\}$.

In other words, (8) and (9) imply that for each $v_t \in V_t$ the values of $i_1, \ldots, i_m, j_1, \ldots, j_m$ are such that if $i_k \neq 0$ then $J(\pi)$ contains some job having its head in M_k, machine assignment X_{i_k}, requirements Y_{j_k} and initial time $\leq t$. For each $k \in \alpha(v_t)$, select exactly one such job and let $H(v_t)$ be the set of heads of the $|\alpha(v_t)|$ jobs so selected. Then (10) asures that the operations of $H(v_t)$ together satisfy all resource constraints. Therefore $H(v_t)$ is a feasible start of π at the initial time t. Conversely, each $H \in H^*$ translates into some $v_t \in V_t$. Moreover,

Lemma 4: For any subproblem π with initial time t, $V_t(\pi)$ and $H_I^c(\pi)$ are isomorphic.

Recurrences (4) and (5) can now be rewritten to reflect the actual computations to be performed.

$$\lambda(\pi,t) = \begin{cases} 0, \text{ if } J(\pi)=\phi. \text{ Otherwise} \\ \min\limits_{v_t \in V_t} \{\max\{t+1, \lambda(\pi(H(v_t)), t(v_t))\}\} \end{cases} \ldots (11)$$

$$\tau(\pi,t) = \begin{cases} 0, \text{ if } J(\pi)=\phi. \text{ Otherwise} \\ \min\limits_{v_t \in V_t} \{(t+1).|T(v_t)| + \tau(\pi(H(v_t)), t(v_t))\} \end{cases} \ldots (12)$$

where $t(v_t) = \max\{t+1, r_{\min}(\pi(H(v_t)))\}$ and

$T(v_t) = \{k \in \alpha(v_t) \mid f_{i_k} = 1\}$.

The algorithms can finally be described. Given π_0, construct X, Y and $S(\pi_0)$. For each considered computation of

$\lambda(\pi,t)$ or $\tau(\pi,t)$, verify if some previously solved subproblem π' with initial time t' satisfies $S(\pi)=S(\pi')$ and $t=t'$. If yes, the answer for both subproblems is the same. Otherwise, calculate $\lambda(\pi,t)$ or $\tau(\pi,t)$, respectively using (11) or (12). The computation stops when $\lambda(\pi_0, r_{min}(\pi_0))$ or $\tau(\pi_0, r_{min}(\pi_0))$ is evaluated.

Sets X and Y can be easily constructed in constant time, while the profile of a subproblem π can be found in $0(n)$ time using (6). In order to compute $\lambda(\pi,t)$ or $\tau(\pi,t)$ by (11) or (12), we need to generate set V_t. This can be done in $0(n)$ time through finding $S_t(\pi)$ using (7) and then applying (8) - (10). For each $v_t \in V_t$, $\pi(H(v_t))$ can be determined from π in constant time. For a fixed t, it is sufficient to compute (11) or (12) for a subset of subproblems which is a cover for π. Therefore the total number of subproblems to be considered is $|\pi_0^c| = 0(n^a)$, $a = |X|.|Y|$. There are $0(n)$ possible initial times. For each newly considered subproblem we need to check whether a previously computed case has the same profile and initial time as it. This can be accomplished by examining all $0(n^a)$ distinct profiles of the subproblems of π_0^c, using linear search. Therefore both algorithms can be implemented in $0(n^{2a+1})$ time and $0(n^a)$ space.

The time bound can be decreased by replacing the linear for binary search. All we need is a total ordering relating the subproblems of the cover π_0^c. For instance, let π', $\pi'' \in \pi_0^c$ and (i,j) be the least pair in the lexicographical ordering of $Z^+_{|X|} \times Z^+_{|Y|}$ such that the element (i,j) of $S(\pi')$ is different from that of $S(\pi'')$. Then define $\pi' < \pi''$ iff s(i,j) is smaller in $S(\pi')$ than in $S(\pi'')$. We can then detect subproblems having identical profiles using binary search. This decreases the time bound to $0(n^{a+2})$.

The minimizing $v_t \in V_t$ defines the actual schedule at time t, as $H(v_t)$ contains respectively the operations to be scheduled in M_k, $k \in \alpha(v_t)$. Otherwise, if $k \notin \alpha(v_t)$ then M_k is left idle.

4. CONCLUSIONS

We have described algorithms for minimizing the length or the mean completion time of a schedule for a job shop problem with unit time operations under resource constraints and release dates. If the number of machines, resources, operations per job and distinct requirements are all fixed then both algorithms terminate within polynomial time, although the degree of the polynomials and the constants involved in the complexity expression grow fast. However, these results could be of theoretical interest due to the generality of the considered problems.

Similar algorithms can be formulated when replacing the release dates by deadlines. It would be interesting to know the complexity of the case when both release dates and deadlines are present together.

REFERENCES

[1] J. Blazewicz, W. Kubiak, H. Rock and J.L. Szwarcfiter, Flow shop scheduling under resource constraints, to appear.

[2] J. Blazewicz, J.K. Lenstra and A.H.G. Rinnooy Kan, Scheduling subject to resource constraints: classification and complexity, Discrete Appl. Math. 5 (1983) 11-24.

[3] N. Hefetz and I. Adiri, An efficient optimal algorithm for the two-machines unit-time jobshop schedule-length problem, Math. Oper. Res. 7 (1982) 354-360.

[4] J.K. Lenstra and A.H.G. Rinnooy Kan, Computational complexity of discrete optimization problems, Ann. Discrete Math.4 (1979) 121-140.

[5] H. Rock, Some new results in flow shop scheduling, Zeits. Oper. Res. 28 (1984) 1-16.