ELSEVIER

European Journal of Operational Research 102 (1997) 502-512

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

One-machine job-scheduling with non-constant capacity – Minimizing weighted completion times

H.F. Amaddeo [a], W.M. Nawijn [b,*], A. van Harten [c]

[a] *KLM Royal Dutch Airlines, Amstelveen, The Netherlands*

[b] *Faculty of Applied Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands*

[c] *Faculty of Technology and Management, University of Twente, The Netherlands*

Received 1 October 1995; accepted 1 July 1996

Abstract

In this paper an n-job one-machine scheduling problem is considered, in which the machine capacity is time-dependent and jobs are characterized by their work content. The objective is to minimize the sum of weighted completion times. A necessary optimality condition is presented and we discuss some special cases where this condition is also sufficient. We prove that the problem is NP-complete. A branch-and-bound algorithm is developed for the case when the capacity function is a step function. Computational results for 1000 test problems are presented. © 1997 Elsevier Science B.V.

Keywords: One-machine scheduling; Non-constant capacity; Weighted completion times

1. Introduction

There are situations in production and service environments in which the bottleneck resource (e.g. a machine) has non-constant capacity in time. For example, labor force may be varying as a consequence of holidays, shifts or breaks. Another example is a machine that has non-constant capacity in time as a consequence of maintenance actions, warming-up or cooling down.

A similar situation occurs at KLM Baggage Handling, where capacity (the number of workers) is non-constant as a consequence of shifts and breaks [2]. As we assume in this paper, the job-progression of a job (the luggage from a flight with destination Amsterdam) in this context is proportional to assigned capacity, and the objective is to minimize the sum of weighted completion times. The former means that the marginal reduction of the work content of a job that is being processed, is proportional to the capacity assigned to that job. By minimizing weighted completion times, different weights can be assigned to different jobs (e.g. luggage from European flights, luggage from intercontinental flights or luggage from delayed flights) consistent with their relative priority.

The problem can also viewed as scheduling a set of jobs (given by their work content) on a single processor when the production rate (the capacity) varies over time, see [4].

It is well known [3] that the problem of minimizing the sum of weighted completion times on a single machine with constant capacity can be solved by sequencing the jobs in non-increasing order of the ratio of weight to processing time.

* E-mail: w.m.nawijn@math.utwente.nl.

PII S0377-2217(96)00240-8

Scheduling jobs in situations with time dependent capacity minimizing weighted completion times is not a trivial case as shown by Baker and Nuttle [1]. They conjectured this problem to be NP-complete and left open the question how to exploit the structure of the problem to obtain optimal solutions. They also found that the well known results of the corresponding single-machine problem with constant capacity could be applied with little or no modification for other functions of the completion times. For the problem at hand, Surkis and Dogramaci [4] developed a myopic heuristic, based on a steepest imminent slope rule (SIS-rule). Applying this rule means that at a certain time the next job scheduled is the job that maximizes the ratio of weight to processing time, which in this case is time-dependent. A linear programming model was formulated to obtain a lower bound on the minimum value of the objective function. They found, using 1000 test problems, that Smith's rule performed better than their SIS-rule.

The content of the paper is as follows. In Section 2 we present the formal problem description, the notations we use in this paper and the assumptions we make. In Section 3, we analyse the general case in which the capacity function is an integrable and bounded function. We first reduce the solution space and second, we deduce a Generalised Local Interchange rule (GLI-rule) which states in what cases two consecutive jobs have to be interchanged to improve the permutation schedule. Third, we investigate sufficient conditions for the local ordering to be optimal. Last, we deduce some optimization results that depend on monotonicity properties. In Section 4, we consider the case where the capacity dynamics can be represented by step functions. In this section we investigate whether more properties of the optimal solution can be derived for this special case and we prove that the scheduling problem is NP-complete. In Section 5, we present a branch&bound-procedure for the case where the capacity function can be presented by a step function. A heuristic solution method is developed based on a combination of Smith's rule and the GLI-rule. In this section we also present computational results of solving 1000 test problems with the branch&bound-procedure using different lower bounds and both heuristic solution methods.

2. Problem definition, notations and assumptions

The set of independent jobs to be scheduled are denoted by $J = \{1, \ldots, n\}$. All jobs are available at time 0. Every job is assigned a weight w_i and needs an amount of work Q_i. We assume without loss of generality that the jobs are ordered by non-increasing ratio of weight to work content (w_i/Q_i). Observe that this ordering essentially coincides with Smith's rule in the constant capacity case.

The capacity of the resource, i.e. the machine, is given by an integrable bounded function $m : \mathbb{R}_+ \to \mathbb{R}_+$ (where $\mathbb{R}_+ = [0, \infty)$) of time. We consider the situation in which the capacity is divisible and can be shared among jobs. The capacity assigned to job i is denoted by the function m_i, which has the same properties as m. The function m can be interpreted as the maximum production rate of the machine and m_i as the production rate of the machine assigned to job i.

For the existence of a feasible solution, the total amount of work to be processed must be less than or equal to the total amount of work that can be handled by the machine, so

$$\sum_{i \in J} Q_i \leq \int_0^\infty m(t)\, \mathrm{d}t.$$

Assuming additivity of job progression as a function of capacity assignment and taking units to measure Q_i compatible with capacity measurements, the following requirements for a solution can be formulated. The capacity assigned to job i should match the amount of work necessary for that job, hence

$$\int_0^\infty m_i(t)\, \mathrm{d}t = Q_i.$$

At any time t, the capacity assigned to the jobs is always less than or equal to the total capacity available, giving

$$\sum_{i \in J} m_i(t) \leq m(t), \quad \text{for all } t \in \mathbb{R}_+.$$

A solution that satisfies the above requirements will be called a *feasible* solution.

Stated this way, our optimization problem is in fact a capacity assignment problem. The objective value of

an assignment $\alpha = (m_1, \ldots, m_n)$ will be denoted by $V(\alpha)$. Given the functions $m_1, m_2, \ldots, m_n$, let

$$C_i = sup\{t \geq 0 \mid m_i(t) > 0\},$$

be the completion time of job i, then $V(\alpha)$ can be expressed as

$$V(\alpha) = \sum_{i \in J} w_i C_i.$$

3. The general case

We start our analysis by showing that it is sufficient to look for an optimal solution within the class of permutation schedules in which jobs are processed one by one. This result is established analogously to other well known reductions of solution spaces of capacity assignment problems.

Theorem 1. *There exists an optimal assignment α^* with the following properties*:

(i) **full capacity utilisation**, *i.e.* $\sum_{i=1}^{n} m_i(t) = m(t)$, *for all* $t \in [0, R]$, *where* R *is such that*

$$\sum_{i \in J} Q_i = \int_0^R m(t)\, dt.$$

(ii) **no concurrency**, *i.e. if* $m_k(t) > 0$, *for* $t \in [t_1, t_2]$, *then* $m_i(t) = 0$, *for* $i \neq k$ *and* $t \in [t_1, t_2]$.

(iii) **no preemption**, *i.e. if job k starts at time* s_k *and is completed at time* C_k, *with*

$$s_k = \inf\{t \geq 0 \mid m_k(t) > 0\},$$

$$C_k = \sup\{t \geq 0 \mid m_k(t) > 0\},$$

then $m_k(t) = m(t)$, *for all* $t \in (s_k, C_k)$.

Proof.

(i) Suppose a capacity assignment does not use full capacity in a time interval $[t, t + \delta]$. Then any job, k say, completed after time t can be finished earlier by assigning (part of) the unused capacity to k, without delaying the completion of other jobs.

(ii) Assume that under assignment α^0, two jobs, say k and l, are simultaneously processed during $[t_1, t_2]$. Suppose job k is processed in $[t_1, t_2]$ prior to job l under an assignment α^*, then the completion time of job l remains the same while the completion time of job k is less than or equal to that under α^0, thus $V(\alpha^*) \leq V(\alpha^0)$.

(iii) Suppose job k is preempted under assignment α^0. Let job k, which starts at time s_k^0, be preempted at time p_k^0, resumed at time r_k^0 and be completed at time C_k^0. Now two situations can occur:

(a) There is a job, say l, completed in the interval $[p_k^0, r_k^0]$. In this case let an assignment α^* be such that the work performed on job k in $[p_k^0, p_k^0]$ under assignment α^0 directly precedes time r_k^0, so no preemption occurs and work performed on other jobs in the preemption period of job k will be performed in the same order directly after time s_k^0. This means that the completion time of job k remains the same and at least one completion time, namely that of job l, decreases. This means that α^* is strictly better than α^0.

(b) No job is completed during the interval $[p_k^0, r_k^0]$. Now α^* is constructed in such a way that the work performed on job k after time r_k^0 under assignment α^0 is performed just after p_k^0, and the work performed on the other jobs in the interval will be completed in the same order just before C_k^0. In doing so the completion time of job k decreases and the rest of the completion times remain the same. Again, α^* is strictly better than α^0. □

Theorem 1 implies that it suffices to consider only non-preemptive capacity assignments, under which only one job is scheduled at a time and full capacity is assigned to each job. This means that an assignment is defined by a permutation of the jobs. Hence, the problem reduces to a sequencing problem. In the sequel a permutation of the jobs is denoted by π and its objective function value by $V(\pi)$.

We will now address the question whether it is possible to generalize Smith's rule to the time dependent capacity case. So let us investigate when an interchange of two consecutive jobs is profitable in the more general case. When the capacity function is a

non-constant integrable function, the switching-rule turns out to be somewhat different. Suppose that job i is started at time A and that the successing job j is completed at time B. We only have to consider the interval $[A, B]$, because all the other completion times remain the same when interchanging jobs i and j. We define the function

$$\xi(t, Q) : D \to \mathbb{R}_+,$$

$$\text{with } D = \{(t, Q) \in \mathbb{R}_+^2 \mid \int_0^t m(x)\,dx \geq Q\},$$

implicitly by

$$\int_{t-\xi}^{t} m(x)\,dx = Q, \quad \text{with } (t, Q) \in D, \tag{1}$$

where we assume that ξ is the smallest such number.

Note that $\xi(t, Q)$ is the processing time needed to finish a job of workload Q at time t using full capacity.

The sum of the values of the weighted completion times of jobs i and j will be denoted by $V(i, j)$, where job i directly precedes job j in a schedule.

To deduce a switching-rule for the interchange of jobs i and j, observe that

$$\begin{aligned}\text{old situation}: V(i,j) &= w_i C_i + w_j C_j \\ &= w_i(B - \xi(B, Q_j)) + w_j B,\\ \text{new situation}: V(j,i) &= w_i C_i^* + w_j C_j^* \\ &= w_i B + w_j(B - \xi(B, Q_i)).\end{aligned}$$

It follows that

$$V(j,i) - V(i,j) = w_i \xi(B, Q_j) - w_j \xi(B, Q_i).$$

For the interchange to be an improvement, this difference has to be negative. Hence, for two adjacent jobs i and j, job j must precede job i in an optimal schedule if

$$\frac{w_j}{\xi(B, Q_j)} > \frac{w_i}{\xi(B, Q_i)}.$$

This criterion will be referred to as the **Generalised Local Interchange rule** (GLI-rule). The criterion does not only depend on the weights and the amounts of work, but also on the capacity function, which enters through the function $\xi(.,.)$. For a constant capacity function this criterion is equivalent to Smith's rule. In general the GLI-rule is a *local* criterion for a given capacity function $m(.)$. Repeated application of this rule may lead to a local optimum, which is not necessarily a global optimum. In the sequel $[i]$ will denote the index corresponding with the job at position i in a schedule. The above analysis implies the following result.

Theorem 2. *A necessary condition for an optimal schedule is that for* $i = 1, \ldots, n-1$

$$\frac{w_{[i]}}{\xi(C_{[i+1]}, Q_{[i]})} \geq \frac{w_{[i+1]}}{\xi(C_{[i+1]}, Q_{[i+1]})}.$$

As mentioned in the introduction, Smith's rule does not guarantee an optimal solution. The reason is that it generally does not generate an ordering which satisfies the necessary condition given in the above theorem as shown by the following example.

Example 3. Consider a problem with capacity function

$$m(t) = \begin{cases} 4, & \text{for } 0 \leq t \leq 3/2, \\ 1, & \text{for } t > 3/2 \end{cases}$$

and three jobs: $w_1 = 4, Q_1 = 4, w_2 = w, Q_2 = 3$, $w_3 = 2, Q_3 = 2$. It is readily verified that if $w < 3$, then the Smith-orderings (3,1,2) and (1,3,2) are optimal. The ordering (1,2,3) is locally optimal (i.e. satisfies Theorem 2) if $0 \leq w \leq 3\frac{3}{7}$. It is globally optimal, however, only if $3 \leq w \leq 3\frac{3}{7}$. Note that it does not satisfy Smith's rule. If $w > 3\frac{3}{7}$, then the Smith-ordering (2,1,3) is the unique optimal solution. Although the ordering (2,3,1) in this case also satisfies Smith's rule, it does not satisfy the necessary condition stated in Theorem 2.

We will now treat some cases in which the application of the GLI-criterion results in an optimal ordering.

Lemma 4. *The only positive continuous functions* $m(t)$, *for which the ratio* $\xi(t, Q_1)/\xi(t, Q_2)$ *is independent of* t, *for all* $Q_1, Q_2 > 0$, *are of the form*

$$m(t) = \frac{\alpha}{1 + \beta t},$$

where $\alpha > 0$. *If* $\beta < 0$, *then the domain of the function is restricted to* $[0, -1/\beta)$.

Proof. As preliminary, notice that by differentiating relation (1) with respect to t, we obtain

$$\frac{\partial \xi(t,Q)}{\partial t} = \frac{m(t-\xi)-m(t)}{m(t-\xi)}. \tag{2}$$

Now, the basic observation is that the following should hold

$$\frac{\partial}{\partial t}\left(\frac{\xi(t,Q_1)}{\xi(t,Q_2)}\right) = \frac{\xi(t,Q_1)}{\xi(t,Q_2)}\left(\frac{1}{\xi(t,Q_1)}\frac{\partial \xi(t,Q_1)}{\partial t} - \frac{1}{\xi(t,Q_2)}\frac{\partial \xi(t,Q_2)}{\partial t}\right) = 0, \tag{3}$$

for all $Q_1, Q_2 > 0$. Hence, it follows from Eqs. (2) and (3) that

$$\frac{1}{\xi(t,Q)}\frac{\partial \xi(t,Q)}{\partial t} = \frac{m(t-\xi)-m(t)}{\xi m(t-\xi)} \tag{4}$$

is independent of Q for all t. Since $\xi(t,Q)$ is strictly increasing in Q the right hand side of Eq. (4) must be independent of ξ.

Now choosing $\xi = t - t_1$ and $\xi = t - t_2$, with $t_1 < t_2$, gives

$$\frac{m(t_1)-m(t)}{(t-t_1)m(t_1)} = \frac{m(t_2)-m(t)}{(t-t_2)m(t_2)}.$$

From this equality, we get

$$m(t) = \frac{m(t_1)m(t_2)(t_2-t_1)}{t(m(t_1)-m(t_2)) + t_2 m(t_2) - t_1 m(t_1)}.$$

Since we only consider functions that are bounded on finite intervals, the only continuous functions that are permitted, are hyperbolic functions of the form

$$m(t) = \frac{\alpha}{1+\beta t}, \quad \text{with } \alpha > 0.$$

Observe that if $\beta = 0$ the case with constant capacity is found. □

The fact that for hyperbolic functions as in Lemma 4 the ratio $\xi(t,Q_1)/\xi(t,Q_2)$ is independent of t, for all Q_1, Q_2, means that in this case the interchange decision for two consecutive jobs implied by the GLI-rule, is independent of time. The optimal ordering for the hyperbolic case is given in the next theorem.

Theorem 5. *If $m(t) = \alpha/(1+\beta t)$, with $\alpha > 0$, $\beta > 0$, then the ordering*

$$\frac{w_{[1]}}{1-e^{-\frac{\beta}{\alpha}Q_{[1]}}} \geq \frac{w_{[2]}}{1-e^{-\frac{\beta}{\alpha}Q_{[2]}}} \geq \cdots \geq \frac{w_{[n]}}{1-e^{-\frac{\beta}{\alpha}Q_{[n]}}}$$

is optimal. If $\beta < 0$ and $m(t)$ defined for $t \in [0, -1/\beta)$, then the inequalities should be reversed.

Proof. From the definition of $\xi(t,Q)$ and $m(t)$ it is easily verified that

$$\xi(t,Q) = (t+\frac{1}{\beta})\{1-e^{-(\alpha/\beta)Q}\}.$$

The GLI-rule gives the ordering stated above. Optimality of this ordering is easily proved by the usual interchange argument using Lemma 4. In fact any ordering that does not satisfy the given ordering can be improved. □

Remark 6. The ordering given in the previous theorem is also optimal for a sequencing problem with constant capacity in which the objective is to minimize

$$\sum_{i=1}^{n} w_{[i]} \exp\left\{\frac{\beta}{\alpha}F_{[i]}\right\}, \quad \text{with } F_{[i]} = \sum_{j=1}^{i} Q_{[j]}.$$

The following optimization results depend on monotonicity properties.

Theorem 7. *If there exists a permutation π^0, for which $w_{[1]} \geq w_{[2]} \geq \cdots \geq w_{[n]}$ and $Q_{[1]} \leq Q_{[2]} \leq \cdots \leq Q_{[n]}$, then the job sequence π^0 is optimal.*

Proof. Since $\xi(t,Q)$ is strictly increasing in Q for every t, it follows that for π^0

$$\frac{w_{[i]}}{\xi(t,Q_{[i]})} \geq \frac{w_{[i+1]}}{\xi(t,Q_{[i+1]})},$$

for all t. Thus, the usual proof, that any permutation different from π^0 can be improved, can be applied. □

Theorem 8. *If there exists a schedule π^0 for which $w_{[i]}/Q_{[i]} \geq w_{[i+1]}/Q_{[i+1]}$ and $Q_{[i]} \geq Q_{[i+1]}$, then π^0 is optimal if m is non-increasing.*

Proof. Let us introduce the shorthand notation $\xi_{[i]} = \xi(t,Q_{[i]})$. First observe that if $Q_{[i]} \geq Q_{[i+1]}$, then

$$\frac{Q_{[i]}}{Q_{[i+1]}} = \frac{\int_{t-\xi_{[i]}}^{t} m(x)\,dx}{\int_{t-\xi_{[i+1]}}^{t} m(x)\,dx} = 1 + \frac{\int_{t-\xi_{[i]}}^{t-\xi_{[i+1]}} m(x)\,dx}{\int_{t-\xi_{[i+1]}}^{t} m(x)\,dx},$$

since $\xi_{[i]} \geq \xi_{[i+1]}$. If $m(.)$, is non-increasing, then

$$\int_{t-\xi_{[i]}}^{t-\xi_{[i+1]}} m(x)\,dx \geq (\xi_{[i]} - \xi_{[i+1]})m(t - \xi_{[i+1]})$$

and

$$\int_{t-\xi_{[i+1]}}^{t} m(x)\,dx \leq \xi_{[i+1]}m(t - \xi_{[i+1]}).$$

Hence, if $Q_{[i]} \geq Q_{[i+1]}$, then

$$\frac{Q_{[i]}}{Q_{[i+1]}} \geq 1 + \frac{(\xi_{[i]} - \xi_{[i+1]})}{\xi_{[i+1]}} = \frac{\xi_{[i]}}{\xi_{[i+1]}} = \frac{\xi(t, Q_{[i]})}{\xi(t, Q_{[i+1]})},$$

for all t. Consequently,

$$\frac{Q_{[i]}}{\xi_{[i]}} \geq \frac{Q_{[i+1]}}{\xi_{[i+1]}},$$

and multiplying the left hand side with $w_{[i]}/Q_{[i]}$ and the right hand side with $w_{[i+1]}/Q_{[i+1]}$, we find that

$$\frac{w_{[i]}}{\xi_{[i]}} \geq \frac{w_{[i+1]}}{\xi_{[i+1]}},$$

for all t.

So again, we have a globally optimal ordering. □

Theorem 9. *If there exists a schedule π^0 for which $w_{[i]}/Q_{[i]} \geq w_{[i+1]}/Q_{[i+1]}$ and $Q_{[i]} \leq Q_{[i+1]}$, then schedule π^0 is optimal if m is non-decreasing.*

Proof. Analogous to the proof of Theorem 8. □

Some corollaries to Theorem 8 and 9 with respect to step functions will be derived in the next section.

Let $[[i]]$ denote the position of job i in an optimal sequence.

Theorem 10. *If $(w_i \geq w_j$ and $Q_i < Q_j)$ or $(w_i > w_j$ and $Q_i \leq Q_j)$, then $[[i]] < [[j]]$.*

Proof. The proof easily follows by an interchange argument. □

The essence of this theorem is its usefulness in restricting the solution space. It will be applied in the branch-and-bound algorithm presented in Section 5.

4. Special case: step functions

In this section we consider capacity functions $m(t)$ that are *step functions*. In production and service environments the capacity function can very often be described by a step function. At KLM Baggage Services the capacity dynamics is modelled this way due to shifts and breaks with a variable number of operators. We define a step function $m : \mathbb{R}_+ \to \mathbb{R}_+$ in the usual way:

$$m(t) = M_i, \quad \text{for } t \in I_i = [b_i, e_i\rangle,$$

with $1 \leq i \leq k$, and $\bigcup_{i=1}^{k} I_i = \mathbb{R}_+$.

Of course, the general results deduced in Section 3 are also valid here.

Let us now investigate the consequences of the general ordering-rule, given in Theorem 2, for step functions. The next lemma states an obvious ordering property within an interval of constant capacity.

Lemma 11. *Let $m : \mathbb{R}_+ \to \mathbb{R}_+$ be a step function and π^0 be a schedule with jobs $[i], \ldots, [i+\ell]$ scheduled within interval I_j, i.e.*

$$s_{[i]} \geq b_j \quad \text{and} \quad C_{[i+\ell]} \leq e_j.$$

If π^0 is optimal then

$$\frac{w_{[m]}}{Q_{[m]}} \geq \frac{w_{[m+1]}}{Q_{[m+1]}}, \quad \text{with } i \leq m \leq i + l - 1.$$

Proof. Assume job $[m]$ and $[m+1]$ are such that $w_{[m]}/Q[m] \leq w_{[m+1]}/Q_{[m+1]}$. Interchanging the jobs will improve the objective function and schedule π^0 is clearly not optimal. □

So the ordering within an interval I_j, with $j = 1, \ldots, k$, must satisfy Smith's rule. One should notice, however, that the question how to assign the jobs to the various intervals is still open.

Let us consider the situation where only one jump of $m(.)$ occurs during the processing of a job, say job $[j]$. We will deduce some ordering-rules between that job and its predecessor, job $[j-1]$, and its successor, job $[j+1]$, assuming that m is constant during the processing of jobs $[j-1]$ and $[j+1]$.

The following two cases are possible

case A : m decreases during job $[j]$ from M_{l-1} to M_l,

case B : m decreases during job $[j]$ from M_{l-1} to M_l.

Let us first consider case A.

Corollary 12 (to Theorem 8).

(i) *If* $w_{[j-1]}/Q[j-1] \geq w_{[j]}/Q_{[j]}$ *and* $Q_{[j-1]} \geq Q_{[j]}$, *then jobs* $[j-1]$ *and* $[j]$ *are ordered according to the GLI-rule.*

(ii) *If* $w_{[j]}/Q[j] \geq w_{[j+1]}/Q_{[j+1]}$, *then jobs* $[j]$ *and* $[j+1]$ *are ordered according to the GLI-rule.*

Proof. For part (i) and (ii) with $Q_{[j]} > Q_{[j+1]}$, the assertion follows from Theorem 8.

We are left with the case $Q_{[j]} < Q_{[j+1]}$ in part (ii). The GLI-rule implies that jobs $[j]$ and $[j+1]$ are scheduled locally optimal if

$$\frac{w_{[j]}}{\xi(t, Q_{[j]})} \geq \frac{w_{[j+1]}}{\xi(t, Q_{[j+1]})}, \quad \text{with } t = C_{[j+1]}.$$

Since $w_{[j]}/Q_{[j]} \geq w_{[j+1]}/Q_{[j+1]}$, $Q_{[j]} < Q_{[j+1]}$, and $t = C_{j+1}$

$$\frac{w_{[j]}}{\xi(t, Q_{[j]})} = \frac{w_{[j]}}{Q_{[j]}/M_l} \geq \frac{w_{[j+1]}}{Q_{[j+1]}/M_l} = \frac{w_{[j+1]}}{\xi(t, Q_{[j+1]})}. \qquad \square$$

In the other situations with m decreasing during the processing of job $[j]$, local optimality of the ordering of two jobs has to be checked explicitly using the GLI-criterion, because no general conclusion can be made beforehand. In these cases optimality does not only depend on the weight and workload of the jobs, but also on the magnitude of the jump and the location of the jobs.

In case B, where m increases during job $[j]$, a similar result holds.

Corollary 13 (to Theorem 9).

(i) *If* $w_{[j-1]}/Q_{[j-1]} \geq w_{[j]}/Q_{[j]} \geq w_{[j+1]}/Q_{[j+1]}$ *and* $Q_{[j-1]} \geq Q_{[j]} \geq Q_{[j+1]}$, *then jobs* $[j-1]$, $[j]$ *and* $[j+1]$ *are ordered according to the GLI-rule.*

(ii) *If* $w_{[j]}/Q_{[j]} \geq w_{[j+1]}/Q_{[j+1]}$ *and* $Q_{[j]}, Q_{[j+1]}$ *are such that, when jobs* $[j]$ *and* $[j+1]$ *are interchanged, job* $[j+1]$ *also overlaps the jump, then jobs* $[j]$ *and* $[j+1]$ *are ordered according to the GLI-rule.*

Proof. The proof is analogous to that of Corollary 12. $\square$

Again, for the other cases left, the GLI-criterion has to be evaluated explicitly to see if jobs are ordered locally optimal. So, although some properties of the optimal solution are known at a jump of the capacity function, no global optimal ordering can be deduced as can also be inferred from the next theorem.

Theorem 14. *For the 1-machine n-job scheduling problem with time dependent machine-capacity, the problem of the minimization of the sum of weighted completion times is* NP-*hard.*

Proof. The proof is established by a transformation from PARTITION.

Let $a_1, \ldots, a_n$ be an instance of PARTITION.

Consider the following instance of our scheduling problem

$$Q_i = w_i = a_i, \quad i = 1, 2, \ldots, n,$$

and let the capacity function m be given by

$$m(t) = \begin{cases} 0, & \text{for } B \leq t \leq B+1, \\ 1, & \text{for } 0 \leq t \leq B \\ & \text{and } B+1 \leq t \leq 2B+1, \end{cases}$$

where $B = 1/2 \sum_{i=1}^{n} a_i$.

Given a schedule π, let $j(\pi)$ be the job with highest index, such that $C_{j(\pi)} \leq B$. If $[i]$ denotes the ith job in the schedule, then

$$\sum_{i=1}^{n} w_{[i]} C_{[i]} = \sum_{i=1}^{n} a_{[i]} \sum_{j=1}^{i} a_{[j]} + \sum_{i=j(\pi)+1}^{n} a_{[i]}.$$

Since the first term on the right hand side is independent of the ordering π (Smith's rule for $w_i = Q_i$), it is clear that PARTITION has a solution if and only if the scheduling problem has a solution with

$$\sum_{i=1}^{n} w_{[i]} C_{[i]} = \sum_{i=1}^{n} a_{[i]} \sum_{j=1}^{i} a_{[j]} + B. \qquad \square$$

Observe that the problem instance, considered in the above proof, concerns a capacity function with two jumps. We conjecture that even for the case of a step function with one jump no polynomial time solution algorithm exists.

5. Step functions: a branch&bound-procedure

In this section we present a B&B-*procedure* to solve the scheduling problem for a capacity function, that is represented by a step function. The number of jumps is arbitrary, say k, and so is the number of jobs, say n. No further restrictions are made.

The B&B-tree is designed in such a way, that a *node* at depth m, with $0 \leq m \leq n-1$ contains a *partial schedule* (π_m) of m jobs ($[1], [2], \ldots, [m]$), and π_n is a complete schedule. Consequently, a *branch* from a node in the B&B-tree corresponds to scheduling a job, not scheduled yet, after job $[m]$. Without loss of generality, we will assume that the jobs, not yet scheduled, are ordered by Smith's rule.

We implemented the following *dominance rule.*

(i) While in an optimal schedule the GLI-criterion has to hold (Theorem 2) for every two neighbours $[i]$ and $[i+1]$, we will only branch on a job j at level m of the B&B-tree if $w_{[m]}/\xi(t, Q_{[m]}) \geq w_j/\xi(t, Q_j)$, with $t = C_{[m+1]}$, and

(ii) applying Theorem 10, we will only branch on a job j at level m of the B&B-tree if there is no job $l (l \neq j)$, not scheduled a depth m, for which ($w_l > w_j$ and $Q_l \leq Q_j$) or ($w_l \geq w_j$ and $Q_l < Q_j$).

The B&B-procedure

STEP 1. Use a *heuristic* to initialize the current "best schedule" (π^b) and determine the upper bound $UB^b = V(\pi^b)$.

STEP 2. Start with an empty schedule ($\pi_0 = \phi$) in the root of the B&B-tree.

STEP 3. Select one of the open nodes in the B&B-tree according to a certain *selection rule.* Branch this node if its *lower bound* is smaller than the best current objective function value UB^b, else fathom this node.

If a new node contains a complete schedule π_n, update π^b and set $UB^b = V(\pi^b)$, if $V(\pi_m) < V(\pi^b)$, and close the node.

If not, calculate a lower bound for the best schedule that can still be obtained, given the partial schedule of this node. Repeat this procedure until there are no open nodes left in the B&B-tree.

STEP 4. The schedule π^b is optimal.

The simplest heuristic to use in step 1 is Smith's rule, that is ordering the jobs by non-increasing ratio of weight to work (w_i/Q_i). We found that the deviation from the optimal value in using Smith's rule was 0.1% on average over 1000 test problems while the average maximum deviation was 0.94%. (The construction of the test problems will be discussed below).

Since Smith's rule does not in general satisfy the GLI-criterion, it might be possible to improve the corresponding schedule by interchanging neighbouring jobs, going from head to tail in the schedule several times until all neighbouring jobs satisfy the GLI-criterion. This GLI-Smith heuristic improved the Smith-schedule in 74.6% of the 1000 test problems and proved to be optimal in 48.6% of the problems. It turned out that the deviation from the optimal value for this improved heuristic was on average 0.04% and the average maximum deviation was 0.75%.

There are several possible selection rules to select the open nodes left in the B&B-tree. We implemented a *best deepest first* selection rule, which means that in the B&B-tree the open node at the deepest level with the smallest lower bound is selected.

Designing efficient lower bounds, necessary to reduce the number of nodes in the tree and to cut down computation time, is not easy in this case, because of the structure of the problem. The following lemma inspired our lower bounds.

Lemma 15. *Consider a problem with capacity function m^0. A lower bound for the optimal solution is provided by solving the problem for a capacity function m^* with the following property*

$$\int_0^t m^0(u)\,du \leq \int_0^t m^*(u)\,du, \quad \text{for all } t \geq 0. \tag{5}$$

Proof. It is easy to see that for every capacity function m^* with the above property, the corresponding problem provides a lower bound to the original prob-

lem, as the completion times under the new capacity function are less than or equal to the completion times under m^0 for every given schedule. This is due to the fact that under the new capacity function at least as much work has been done at any point in time. □

Two lower bounds easily follow from this lemma. The simplest lower bound ($LB1$) is provided by using the constant capacity function defined by

$$M^* = \max_{i=1,\ldots,k} M_i^0,$$

which equals the maximum offered production rate.

The second lower bound ($LB2$) we implemented is more sophisticated. Assume that

$$\mathcal{M}^0(t) = \int_0^t m^0(u)\,du.$$

The constant capacity function that provides the second lower bound is constructed in the following way

$$M^* = \max_{i=1,\ldots,k} \frac{\mathcal{M}^0(e_i)}{e_i}.$$

M^* can be interpreted as the maximum time-averaged production rate.

The fact that both capacity functions, mentioned above, satisfy Lemma 15 and that $LB2$ is a sharper lower bound than $LB1$ can be easily seen graphically.

Lower bounds based on non-constant capacity functions satisfying Lemma 15 are more difficult to implement, because of the complexity of the resulting scheduling problem. Nevertheless, we designed a lower bound ($LB3$) based on the following non-constant decreasing capacity function (one jump),

$$m^*(t) = \begin{cases} M_1^*, & \text{for } t < e_1^*, \\ M_2^*, & \text{for } t \geq e_1^*, \end{cases}$$

with e_1^* following from

$$M_1^* e_1^* = \mathcal{M}^0(e_k) - M_2^*(e_k - e_1^*)$$

and

$$M_1^* = \max_{i=1,\ldots,k} \frac{\mathcal{M}^0(e_i)}{e_i},$$

$$M_2^* = \min_{i=1,\ldots,k} \frac{\mathcal{M}^0(e_k) - \mathcal{M}^0(b_i)}{e_k - b_i}.$$

Observe that $m^*(t)$ is the smallest (point wise) capacity function with one jump satisfying (5). Notice that lower bound $LB3$ is at least as sharp as $LB2$.

If $M_1^* = M_2^*$, the resulting capacity function is constant and equals the average capacity, in which case $LB3$ equals $LB2$. If $M_1^* > M_2^*$, we have a scheduling problem with one jump. To obtain a lower bound for this problem we applied Lagrangian Relaxation. We omit the details, however, since the computation time using this lower bound is higher than the computation time using $LB2$ (see Table 1).

Non-constant lower bounds based on hyperbolic capacity functions (Lemma 4) and Lemma 15 can also be constructed. We did not implement such lower bounds, however, because we expected them not to reduce the number of nodes very much, while computation time per node will increase a lot. This increase in computation time is caused by the complex computations that have to be performed to find a hyperbolic function that not only satisfies Lemma 15 but also produces a good lower bound.

We constructed 40 combinations of the number of jobs n and the number of intervals k, namely $n = 20, 24, 28, 32, 36, 38, 40$ combined with $k = 5, \ldots, 9$. For each combination 25 randomly generated problems were solved using the different lower bounds in the B&B-procedure. In this way a total of 1000 test problems were constructed and solved.

The experiments of Surkis and Dogramaci [4] showed that the variation in the difference between the Smith-solution and the optimal solution increases when the variation in w_i and Q_i increases, and also showed that the larger the fluctuation in capacity, the larger the deviation from the optimal solution. Therefore, the widths of the supports of the uniform distributions to be used in the randomly generated test problems, will be taken relatively large. The weight and work content of each job are drawn randomly from a uniform distribution ($U(1,10)$ and $U(2,40)$, respectively). The weight and work content of every job are normalized in such a way that the total amount of work is 1000. This is accomplished by multiplying weight and work content with $1000/\sum_{i \in J} Q_i$. The capacity at every interval and the interval length are also drawn from a uniform distribution ($U(4,30)$ and $U(5,20)$, respectively). The capacity function is also normalized. First, the intervals are normalized

Table 1
Average computational results for different lower bounds

n	$LB1$		$LB2$		$LB3$	
	# nodes	CPU time (sec)	# nodes	CPU time (sec)	# nodes	CPU time (sec)
20	1168	1.90	1027	1.90	1019	2.14
24	3092	6.24	2435	5.42	2402	5.97
28	10967	22.34	9778	21.42	9490	23.37
32	37280	87.79	31858	81.05	31479	90.08
34	38869	106.24	32546	91.66	31707	98.92
36	143738	374.11	119187	336.02	115351	360.79
38	173494	504.76	151754	484.74	150050	521.24
40	248123	797.93	223498	796.41	220852	870.78

Table 2
Average computational results for $LB2$ (# nodes in B&B tree, CPU time (sec))

n	$k=5$	6	7	8	9	
20	558	692	1027	916	1272	(# nodes)
	1.08	1.28	1.90	1.75	2.32	(sec)
24	1390	2771	2436	2560	2791	
	3.18	6.45	5.42	5.58	6.08	
28	5108	3762	9779	22787	6526	
	13.9	10.1	21.4	52.2	17.2	
32	6243	83779*	31858	25989	125848*	
	19.3	244.3*	81.0	70.8	317.3*	
34	10160	20036	32547	58006	80189	
	34.4	61.9	91.7	153.6	205.0	
36	21726	25129	119187	169119	291708*	
	73.8	81.1	336.0	481.2	827.1*	
38	48981*	118950	15174	556230*	251846	
	187.0*	391.8	484.7	2335.2*	700.3	
40	45918	147874	223498	216052	273030	
	181.4	519.1	796.4	643.4	792.0	

such that the total time is 100. Then, the M_i's are normalized such that the total capacity is 1000. The numbers in the tables are the averages of 25 experiments for every combination. From Table 1 one can see, as to be expected, that the number of nodes in the B&B-tree grows exponentially in the number of jobs (n). The number of nodes does not grow that fast in the number of jumps of the capacity function. The latter is due to the fact that in the B&B-procedure the branches are built independently of the number of intervals. The number of jumps does however influence the computation of the lower bounds for the different branches. The average computation time of the marked cases in Table 2 are rather high compared to the computation times of the other combinations. This is due to the fact that the computation time was very long for one or more of the 25 problems solved for that case.

In these cases the number of nodes that had to be investigated were not reduced very much by the dominance rules and our lower bound was not very effective.

6. Conclusions

In this paper a single machine scheduling problem with time-dependent machine capacity was investigated, in which the objective is to minimize the weighted completion times. The problem was shown to be NP-hard.

A necessary optimality condition has been derived that prescribes the ordering between two adjacent jobs. Some specific cases has been discussed in which this condition is also sufficient. Moreover, the condition can be used to improve the Smith schedule. The resulting heuristic proves to be quite good when applied to capacity functions that are step functions.

An B&B algorithm for the case of step functions has been considered and applied to 1000 randomly generated test problems. The best lower bound ($LB2$) from a computational point of view, is obtained from a constant capacity function that equals the maximum time-averaged production rate corresponding to the function $m(t)$. From the computational results on the test problems it followed that the number of nodes in the B&B-tree was quite insensitive with respect to the number of jumps of the capacity function. We finally note that an analogous B&B-procedure can also be implemented for the general case using the same type of lower bounds.

References

[1] Baker, K.R., and Nuttle, H.L.W. (1980), "Sequencing independent jobs with a single resource", *Naval Research Logistics Quarterly* 27/3, 499–510.

[2] Bootsma, P.D., and van Harten, A. (1993), "Predicting Manpower Requirements at KLM Baggage Handling", Working Paper, Department of Management Science and Logistics, University of Twente.

[3] Smith, W.E. (1956), "Various Optimizers for Single-stage Production", *Naval Research Logistics Quarterly* 3/1, 59–66.

[4] Surkis, J. and A. Dogramaci (1988), "Minimizing the Sum of Weighted Completion Times of n-Independent Jobs when Resource Availability Varies over Time: Performance of Simple Priority Rule", *Naval Research Logistics Quarterly* 35/1, 35–47.