Minimizing total weighted completion time in a proportionate flow shop

Natalia Shakhlevich
Academy of Sciences of Belarus
Surganov Street 6, 220012 Minsk, Belarus

Han Hoogeveen
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Michael Pinedo
Department of Industrial Engineering and Operations Research
Columbia University
New York, NY 10027

January 16, 1996

Abstract

We study the special case of the $m$ machine flow shop problem in which the processing time of each operation of job $j$ is equal to $p_j$; this variant of the flow shop problem is known as the proportionate flow shop problem. We show that for any number of machines and for any regular performance criterion we can restrict our search for an optimal schedule to permutation schedules. Moreover, we show that the problem of minimizing total weighted completion time is solvable in $O(n^2)$ time.


Key Words and Phrases: flow shop scheduling, weighted completion time.
1 Introduction

In many manufacturing and assembly facilities a number of operations have to be done on every job. Often, these operations have to be done on all jobs in the same order, which implies that the jobs have to follow the same route through the machines. Such an environment is referred to as a flow shop. Mathematically, the flow shop model is described as follows. There are $m$ machines $M_i$ ($i = 1, \ldots, m$) that have to process $n$ jobs $J_j$ ($j = 1, \ldots, n$). The machines are continuously available from time zero onwards and can handle at most one job at a time. Each job $J_j$ consists of a chain of $m$ operations $O_{ij}$ ($i = 1, \ldots, m; j = 1, \ldots, n$), which implies that the execution of $O_{ij}$ cannot start before the execution of $O_{i-1,j}$ has been completed, for $i = 2, \ldots, m$. Operation $O_{ij}$ has to be processed by $M_i$, and its processing requires an uninterrupted period of length $p_{ij}$, which, without loss of generality, we assume to be integral. We further assume that there is an unlimited buffer space in between two machines.

A schedule is defined as a set of completion times $C_{ij}$ ($i = 1, \ldots, m; j = 1, \ldots, n$) that satisfy the constraints mentioned above. The completion time of job $j$ is defined as $C_j = C_{mj}$, for $j = 1, \ldots, n$. Our objective is to minimize total weighted completion time, that is, we want to find a schedule with minimum $\sum_{j=1}^{n} w_j C_j$ value, where $w_j$ denotes the weight of job $j$ ($j = 1, \ldots, n$), which, without loss of generality, we assume to be a positive integral number.

Garey, Johnson, and Sethi (1976) show that the problem of minimizing total completion time in a two-machine flow shop is already $\mathcal{NP}$-hard in the strong sense. Hence, it is very unlikely that there exists a polynomial algorithm for our general problem. Two special cases have been addressed in the literature. The first one concerns the case in which the matrix of processing times is dominant; Strusevich (1981; see also Tanaev, Sotskov, and Strusevich (1994)) and Van den Nouweland et al. (1992) show that this problem is solvable in $O(nm + n^2)$ time. The second one deals with an ordered matrix of processing times, with the additional constraint that all weights are equal; Panwalker and Khan (1976) show this problem to be solvable in $O(n \log n)$ time.

We consider the special case in which all operations belonging to job $j$ ($j = 1, \ldots, n$) have processing time $p_j$. Such a flow shop is in the literature often referred to as a proportionate flow shop, see Ow (1985) and Pinedo (1985, 1995). This special case can occur for instance when job $j$ corresponds to a customer order with a quantity $q_j$ associated with it. The processing time of this job on any one of the machines is always proportionate to the quantity $q_j$. Actually, a more general (and possibly more appropriate) model would be the model where the processing time of job $j$ on machine $i$ is a function of the quantity $q_j$ as well as the speed $s_i$ of machine $i$, i.e., $p_{ij} = q_j / s_i$.

Pinedo observed that proportionate flow shop models in a number of cases are very similar to their single machine counterparts (see Pinedo (1995), page 103). Often, the rule that solves the single-machine scheduling problem needs only a very slight modification to be applicable to the corresponding proportionate flow shop. This is true with regard
to the following objectives:

- minimizing total completion time;
- minimizing maximum lateness;
- minimizing total tardiness;
- minimizing total number of tardy jobs.

However, with the total weighted completion time objective, the proportionate flow shop turns out to be quite different from the single machine scheduling problem with the same objective. In this paper, we show that this problem is solvable in $O(n^2)$ time.

This paper is organized as follows. In Section 2, we show that the optimal schedule is a permutation schedule, that is, a schedule in which each machine processes the jobs in the same order. In Section 3, we provide a further characterization of the optimal schedule. In Section 4, we show that the problem is solvable in $O(n^2)$ time. In the last section, we draw some conclusions.

2 Optimality of permutation schedules

In this section, we show that any schedule with minimum total weighted completion time value must be a permutation schedule. Moreover, we show that, for any regular objective function, we can restrict ourselves to permutation schedules, where an objective function is called regular if an increase in the completion times cannot decrease the value of the objective function.

Let $\sigma$ denote any schedule, and for ease of exposition, suppose that the jobs are numbered according to the order in which they are executed by the last machine in $\sigma$. It is easily checked that, if the processing order of the jobs is the same on each machine, then

$$C_j(\sigma) = \sum_{i=1}^{j} p_i + (m-1) \max(p_1, \ldots, p_j) \quad \forall \ j = 1, \ldots, n.$$ 

We use this observation in the next lemma in which we show that there is an optimal schedule that is a permutation schedule.

**Lemma 1** Let $\sigma$ be any schedule in which the last machine executes the jobs in the order $J_1, \ldots, J_n$. Then

$$C_j(\sigma) \geq \sum_{i=1}^{j} p_i + (m-1) \max(p_1, \ldots, p_j) \quad \forall \ j = 1, \ldots, n.$$
Proof. We will show that no such schedule $\sigma$ can exist with

$$C_j(\sigma) < \sum_{i=1}^{j} p_i + (m - 1) \max(p_1, \ldots, p_j)$$

for any $j$. We will first show this for the jobs $J_j$ ($j = 1, \ldots, n$) with processing times larger than the processing times of all preceding jobs, that is, $p_j > \max(p_1, \ldots, p_{j-1})$. Let $J_j$ be any such job. Since $\sigma$ is a feasible schedule, the execution of the operations of $J_j$ takes $m p_j$ time. We have to show that the total time that elapses between the completion of $J_j$ at machine $M_{h-1}$ and the start of $J_j$ at machine $M_h$ ($h = 1, \ldots, m$) amounts to at least $\sum_{i=1}^{j-1} p_i$; for reasons of consistency, we assume that $J_j$ is completed at time zero by machine $M_0$.

Consider the transit from machine $M_{h-1}$ to $M_h$ for any $h \geq 1$. Let $I(h)$ denote the set of jobs that pass job $j$ between these two machines. Since $\sigma$ is a feasible schedule, the execution of any job from $I(h)$ by $M_{h-1}$ cannot start until $J_j$ is completed by $M_{h-1}$, and the execution of $J_j$ by $M_h$ cannot start before the last job in $I(h)$ is completed. Hence, if $I(h) \neq \emptyset$, then more than $\sum_{i \in I(h)} p_i$ time elapses between the start of $J_j$ on $M_h$ and the completion of $J_j$ on $M_{h-1}$. Let $I(1)$ define the set of jobs that precede $J_j$ on $M_1$; hence, the time that elapses between the start of job $j$ on $M_1$ and the completion of $J_j$ on $M_0$ is exactly equal to $\sum_{i \in I(1)} p_i$. Since the jobs $J_1, \ldots, J_{j-1}$ precede $J_j$ on $M_m$, we have that

$$I(1) \cup \ldots \cup I(m) \supseteq \{J_1, \ldots, J_{j-1}\},$$

and hence

$$\sum_{h=1}^{m} \sum_{J_j \in I(h)} p_i \geq \sum_{i=1}^{j-1} p_i,$$

where the equality sign is attained only when $I(h) = \emptyset$ for $h = 2, \ldots, m$.

We complete the proof by showing the desired result for the remaining jobs $J_j$, which have $p_j \leq \max(p_1, \ldots, p_{j-1})$. Consider any such job $J_j$; let $J_g$ be the smallest indexed job with processing time equal to $\max(p_1, \ldots, p_j)$. Because of the job-sequence on $M_m$, we have that

$$C_j(\sigma) - C_g(\sigma) \geq \sum_{i=g+1}^{j} p_i.$$ 

As we have just proven that

$$C_g(\sigma) \geq \sum_{i=1}^{g} p_i + (m - 1)p_g,$$

we obtain that

$$C_j(\sigma) \geq \sum_{i=1}^{j} p_i + (m - 1)\max(p_1, \ldots, p_j).$$
Corollary 1 If the objective function is regular, then there exists a permutation schedule that is optimal. If the objective is to minimize total weighted completion time and all processing times and weights are positive, then the optimal schedule is a permutation schedule.

Hence, we can deduce the completion times when we know the sequence in which the last machine executes the jobs. Filling in

\[ C_j = \sum_{i=1}^{j} p_i + (m - 1) \max(p_1, \ldots, p_j), \]

we obtain that

\[ \sum_{j=1}^{n} w_j C_j = \sum_{j=1}^{n} \sum_{i=1}^{j} w_j p_i + (m - 1) \sum_{j=1}^{n} w_j \max(p_1, \ldots, p_j). \]

The first component is exactly equal to the total weighted completion time of \( \sigma \) on a single-machine; this term is minimized by scheduling the jobs in weighted shortest processing time (WSPT) order, that is, in order of nonincreasing \( w_j/p_j \) ratio (Smith, 1956). The second component is minimized by scheduling the jobs in order of nondecreasing processing times. This suggests that if for two jobs \( j \) and \( k \) both \( w_j/p_j \geq w_k/p_k \) and \( p_j \leq p_k \), then \( J_j \) should precede \( J_k \) in an optimal schedule. In the next section, we show that this intuitive dominance rule holds indeed.

3 Characterization of an optimal schedule

We start with some definitions to facilitate wording. We define \( f(\sigma) \) as the cost of schedule \( \sigma \), that is, \( f(\sigma) = \sum_{j=1}^{n} w_j C_j(\sigma) \). Furthermore, we introduce the concept of a new-max job, which is defined as a job with a processing time that exceeds the processing times of all its predecessors in the schedule.

Lemma 2 Let \( J_j \) and \( J_k \) be two jobs for which both \( w_j/p_j \geq w_k/p_k \) and \( p_j \leq p_k \). There exists an optimal schedule in which job \( j \) precedes job \( k \). If \( w_j/p_j > w_k/p_k \), then \( J_j \) precedes \( J_k \) in any optimal schedule.

Proof. Our proof is based upon presenting another optimal schedule that satisfies the conditions of the lemma. Let \( \sigma \) be an optimal schedule that does not satisfy the dominance rule. Hence, we can identify two jobs \( j \) and \( k \) that satisfy the terms of the dominance rule, whereas \( J_j \) succeeds \( J_k \); \( \sigma \) has the form \( (\pi_1, k, \pi_2, j, \pi_3) \), where \( \pi_i \) (\( i = 1, 2, 3 \)) are the corresponding subschedules of \( \sigma \). We specify two alternative schedules \( \sigma_1 \) and \( \sigma_2 \) in which \( J_j \) precedes \( J_k \) and show that one of them is optimal too. Schedule \( \sigma_1 \) is obtained by inserting job \( j \) immediately in front of job \( k \); \( \sigma_2 \) is obtained by inserting job \( k \) immediately
after job $j$. Hence, $\sigma_1 = (\pi_1, j, k, \pi_2, \pi_3)$ and $\sigma_2 = (\pi_1, \pi_2, j, k, \pi_3)$; in both cases, the completion times of the jobs in $\pi_1 \cup \pi_3$ remain unchanged. We compute upper bounds on $f(\sigma) - f(\sigma_1)$ and $f(\sigma) - f(\sigma_2)$ and show that at least one of these has to be nonnegative.

We first compare $\sigma$ and $\sigma_1$. Moving $J_j$ forward to obtain $\sigma_1$ from $\sigma$ does not change the new-max job faced by job $k$ and the jobs in $\pi_2$, as $p_k \geq p_j$. The new-max job faced by $J_j$ may change; let $J_\alpha$ and $J_\beta$ be the new-max jobs faced by $J_j$ in $\sigma$ and $\sigma_1$, respectively. Hence, the completion times of job $k$ and the jobs in $\pi_2$ increase by $p_j$, whereas the completion time of $J_j$ is decreased by $(p_k + p(\pi_2) + (m - 1)(p_\alpha - p_\beta))$, where $p(\pi_2)$ denotes the total processing time of the jobs in $\pi_2$; similarly, $w(\pi_2)$ denotes the total weight of the jobs in $\pi_2$. Filling this in, we obtain

$$f(\sigma) - f(\sigma_1) = w_j[p_k + p(\pi_2) + (m - 1)(p_\alpha - p_\beta)] - p_j(w_k + w(\pi_2)) \equiv \Delta_1.$$ 

Postponing $J_j$ to obtain $\sigma_2$ from $\sigma$ does not increase the size of the processing time of the new-max job faced by job $j$ and the jobs in $\pi_2$; hence, $p_k$ is a lower bound on the decrease of the completion times for these jobs. In $\sigma$, job $k$ either faces $J_\beta$ as a new-max job, or it is a new-max job itself. In $\sigma_2$, job $k$ faces $J_\alpha$ as a new-max job; $J_k$ and $J_\alpha$ can be the same. Hence, the completion time of job $k$ is increased by no more than $p_j + p(\pi_2) + (m - 1)(p_\alpha - p_\beta)$. Filling this in, we obtain

$$f(\sigma) - f(\sigma_2) \leq p_k(w_j + w(\pi_2)) - w_k[p_j + p(\pi_2) + (m - 1)(p_\alpha - p_\beta)] \equiv \Delta_2.$$ 

We now show that at least one of $\Delta_1$ and $\Delta_2$ is nonnegative. Suppose to the contrary that $\Delta_1 < 0$ and $\Delta_2 < 0$. Hence, then also $\Delta \equiv w_k \Delta_1 + w_j \Delta_2 < 0$. Working things out yields (the terms with $(m - 1)$ and $p(\pi_2)$ cancel out):

$$\Delta = w_j w_k p_k - p_j w_k (w_j + w(\pi_2)) + w_j p_k (w_j + w(\pi_2)) - w_j w_k p_j =$$

$$[w_j + w_k + w(\pi_2)](w_j p_k - w_k p_j) \geq 0,$$

as $w_j/p_j \geq w_k/p_k$. This contradiction shows that, either at least one of $\Delta_1$ and $\Delta_2$ is positive, or $\Delta_1 = \Delta_2 = 0$; this last case can occur only if $w_j/p_j = w_k/p_k$. By this, we have shown the second part of the lemma.

What remains to be shown is that through this interchange argument we can obtain an optimal schedule that satisfies the dominance rule even in case of jobs with equal weight to processing time ratios. Hence, we have to indicate an order in which the interchanges should be applied such that we do not end in a cycle.

We have already shown that in each optimal schedule the only jobs that are not scheduled according to the order imposed by the dominance rule have equal weight to processing time ratio. Let $Q_j$ be the set containing all jobs $J_i$ with weight to processing time ratio equal to $w_j/p_j$, and suppose that the jobs in $Q_j$ are not in order of nondecreasing processing times. Let $\bar{Q}_j$ denote the sequence in which the jobs in $Q_j$ appear in the schedule. Let $g$ be the first position in which $\bar{Q}_j$ and the shortest processing time
order differ; suppose that \( J_k \) and \( J_l \) occupy position \( g \) in both sequences. We apply our interchange argument to the jobs \( k \) and \( l \); this either yields a schedule with smaller cost, or a schedule with equal cost in which \( J_l \) is inserted immediately before \( J_k \). If we continue with this schedule and proceed in the same manner, then we will end up with a schedule that satisfies the dominance rule stated in Lemma 2.

We can characterize the form of an optimal schedule even further. To that end, we first introduce some notation that will be used throughout the paper. Recall that the jobs have been renumbered according to the processing order in \( \sigma \) and that job \( j \) is a new-max job in \( \sigma \) if and only if \( p_j > \max(p_1, \ldots, p_{j-1}) \), for \( j = 1, \ldots, n \). We partition \( \sigma \) on basis of these new-max jobs. Let \( J_{c_1}, \ldots, J_{c_r} \) be the new-max jobs in \( \sigma \). Then \( \sigma \) is partitioned into \( r \) so-called segments, the ith of which contains \( J_{c_i} \) and all jobs between \( J_{c_i} \) and \( J_{c_{i+1}} \), where we define \( J_{c_{r+1}} \) to be \( J_n \). We say that the ith segment belongs to job \( c_i \). Hence, if \( J_j \) and \( J_{j+1} \) are part of the same segment in \( \sigma \), then \( C_{j+1} = C_j + p_{j+1} \), as both jobs face the same new-max job.

**Lemma 3** In any optimal schedule the jobs in the same segment, including the new-max job, are in order of nonincreasing \( w_j/p_j \) ratio.

**Proof.** Let \( Q \) be the set of jobs in the segment and let \( J_q \) be the new-max job. Application of the dominance rule of Lemma 2 shows that each job \( j \) in the segment must have \( w_j/p_j \leq w_q/p_q \). Moreover, if \( w_j/p_j = w_q/p_q \), then \( p_j \) must be equal to \( p_q \), as we would gain by interchanging \( J_j \) and \( J_q \) otherwise. Since all jobs \( j \in Q \) face the same new-max job,

\[
C_j = C_q + \sum_{i=q+1}^{j} p_i.
\]

Hence,

\[
\sum_{j \in Q} w_j C_j = C_q \sum_{j \in Q} w_j + \sum_{j \in Q} w_j \sum_{i=q+1}^{j} p_i.
\]

The first term is a constant, and the second term is minimized by scheduling the jobs in \( Q \) in WSPT order, as has been shown by Smith (1956).

4 Finding an optimal schedule in \( O(n^2) \) time

We are now ready to state our \( O(n^2) \) algorithm that minimizes total weighted completion time. This algorithm first orders the jobs according to WSPT and then constructs a schedule progressively taking each time a job from the WSPT sequence and inserting it in the partial schedule in such a way that the incremental cost is minimized. We refer to this algorithm as the WSPT-MCI algorithm.
Algorithm WSPT-MCI

Step 1. Reindex the jobs in WSPT order, settling ties according to nondecreasing processing times. Let $\sigma_1$ be the sequence consisting of $J_1$; set $j \leftarrow 2$.

Step 2. Derive $\sigma_j$ from $\sigma_{j-1}$ by inserting $J_j$ in $\sigma_{j-1}$ such that the cost increase is minimum; if there are several possibilities, then choose the one in which $J_j$ is inserted latest. Set $j \leftarrow j + 1$.

Step 3. If $j \leq n$, then go to Step 2.

Step 4. Determine an optimal permutation schedule by scheduling the jobs in order of occurrence in $\sigma_n$.

Because of Lemma 2, we know that we should not insert $J_j$ before any job $i$ with processing time $p_i \leq p_j$. Moreover, we know from Lemma 3 that we must insert it immediately before a new-max job. Therefore, we can determine the cost of inserting $J_j$ in constant time for each of the $j$ possible positions. Hence, Step 2 takes $O(n)$ time per iteration, which implies that Algorithm WSPT-MCI can be made to run in $O(n^2)$ time.

Note that Algorithm WSPT-MCI is only guaranteed to determine an optimal solution if the jobs are added in WSPT order; it is possible to find an instance for which a greedy algorithm like Algorithm WSPT-MCI finds a non-optimal solution if the jobs are added in another order.

Theorem 1 Algorithm WSPT-MCI yields an optimal schedule.

Proof. We prove Theorem 1 by showing that there exists an optimal schedule in which the first $j$ ($j = 1, \ldots, n$) jobs appear in the same order as in the sequence $\sigma_j$.

The above statement is certainly true for $j = 1$. Let $s^*$ be an optimal schedule that satisfies Lemmas 2 and 3 in which the jobs $J_1, \ldots, J_{j-1}$ are scheduled according to the order in $\sigma_{j-1}$; we further assume that jobs with equal processing time and weight are scheduled in order of increasing index. We will show that, if necessary, we can modify $s^*$ without increasing the cost such that the processing order of the jobs $J_1, \ldots, J_j$ in $s^*$ matches $\sigma_j$, such that Lemmas 2 and 3 hold, and such that equal jobs are scheduled in order of increasing index. Our proof is based on a case-by-case analysis. We distinguish between the characteristics of the possible insertion spots for $J_j$. We consider the following three cases:

(i) $J_j$ is a new-max job in $\sigma_j$;

(ii) $J_j$ is not a new-max in $\sigma_j$ and $J_j$ precedes some job in $\{J_1, \ldots, J_{j-1}\}$ in $s^*$ that it succeeds in $\sigma_j$;

(iii) $J_j$ is not a new-max in $\sigma_j$ and $J_j$ succeeds some job in $\{J_1, \ldots, J_{j-1}\}$ in $s^*$ that it precedes in $\sigma_j$. 

8
Proof of (i). If \( J_j \) is a new-max job in \( \sigma_j \), then all jobs preceding \( J_j \) in \( \sigma_j \) have smaller processing time and, because of Lemma 2, all jobs succeeding \( J_j \) have larger processing time; note that in this case no job in \( \{J_1, \ldots, J_{j-1}\} \) with processing time equal to \( p_j \) can exist, since it should have preceded \( J_j \) then. Since \( s^* \) satisfies Lemma 2, \( J_j \) does not precede any job with smaller processing time that is also present in \( \sigma_j \). What remains to be shown is that in \( s^* \) \( J_j \) precedes the jobs in \( \sigma_{j-1} \) with processing time larger than \( p_j \), or that \( s^* \) can be modified that way without increasing the cost of the schedule. Suppose to the contrary that some lower-indexed job \( q \) with \( p_q > p_j \) precedes \( J_j \) in \( s^* \); if there are more such jobs, then we choose \( J_q \) to be the first one in \( s^* \). The schedule \( s^* \) has then the form \((\pi_A, \pi_B, q, \pi_C, j, \pi_D)\). If we remove all jobs \( J_{j+1}, \ldots, J_n \) from \( s^* \), then we obtain the sequence \( \sigma^* = (\pi_a, \pi_b, q, \pi_c, j, \pi_d) \). Note that \( \sigma_j \) has the form \((\pi_a, j, \pi_b, q, \pi_c, \pi_d)\), since all jobs \( J_1, \ldots, J_{j-1} \) appear in the same order in \( s^* \) as in \( \sigma_{j-1} \) and \( \sigma_j \) is obtained by inserting \( J_j \) in \( \sigma_{j-1} \). We will show that the schedule \( s = (\pi_A, j, \pi_B, q, \pi_C, \pi_D) \), where \( J_j \) is inserted immediately before the first job in \( \pi_b \), has cost no more than \( s^* \). Note that all jobs in \( \pi_b \) have processing time larger than \( p_j \). Hence, \( s \) satisfies Lemmas 2 and 3, as \( J_j \) is moved forward and inserted immediately before a larger job.

To show that \( f(s^*) - f(s) \geq 0 \), we first compute \( f(\sigma^*) - f(\sigma_j) \). Since all jobs in \( \pi_b \) have processing time larger than \( p_j \), all jobs except for \( J_j \) face the same new-max job in \( s \) as in \( s^* \). Let \( J_o \) be the new-max job faced by \( J_j \) in \( \sigma^* \); \( J_j \) is a new-max job itself in \( \sigma_j \). Hence,

\[
f(\sigma^*) - f(\sigma_j) = w_j[p(\pi_b) + p_q + p(\pi_c) + (m - 1)(p_o - p_j)] - p_j[w(\pi_b) + w_q + w(\pi_c)].
\]

Now we compute \( f(s^*) - f(s) \). Due to the choice of moving \( J_j \) forward to the position immediately before the first job from \( \pi_b \) in \( s^* \), no job except for \( J_j \) will face another new-max job. Because of Lemma 2, no job in \( \pi_A \setminus \pi_o \) and \( \pi_C \setminus \pi_c \) can have processing time larger than \( p_j \). Hence, \( J_j \) again faces \( J_o \) as its new-max job in \( s^* \) and is a new-max job itself in \( s \). Therefore,

\[
f(s^*) - f(s) = w_j[p(\pi_B) + p_q + p(\pi_C) + (m - 1)(p_o - p_j)] - p_j[w(\pi_B) + w_q + w(\pi_C)] = f(\sigma^*) - f(\sigma_j) + w_j[p(\pi_b \setminus \pi_b) + p(\pi_c \setminus \pi_c)] - p_j[w(\pi_b \setminus \pi_b) + w(\pi_c \setminus \pi_c)].
\]

Since \( \sigma_j \) and \( \sigma^* \) both can be obtained by inserting \( J_j \) in the sequence \( \sigma_{j-1} \), we have that \( f(\sigma^*) - f(\sigma_j) \geq 0 \), as \( \sigma_j \) is selected as the one with minimal cost. Since each \( J_i \in (\pi_B \setminus \pi_b) \cup (\pi_C \setminus \pi_c) \) has \( w_i/p_i \leq w_j/p_j \), the sum of the second and third term is nonnegative as well.

If \( J_j \) is not a new-max job in \( \sigma_j \), then it must be inserted in \( \sigma_{j-1} \) as the last job in the segment belonging to some new-max job \( J_q \); let \( \pi_d \) denote the other jobs in the segment belonging to \( J_q \) in \( \sigma_j \). We have to show that, if necessary, we can modify \( s^* \) to a schedule \( s \) in which \( J_j \) occupies this position too. Depending on the position of \( J_j \) in \( s^* \), we distinguish between the cases (ii) and (iii).
Proof of (ii). First suppose that $J_j$ is scheduled before $J_q$ in $s^*$. By assumption, we have that the order in which $J_1, \ldots, J_{j-1}$ occur in $s^*$ is the same as in $\sigma_{j-1}$. This assumption, in combination with Lemma 2, implies that in $s^*$ $J_q$ and the jobs in $\pi_d$ are scheduled consecutively. Hence, $s^*$ has the form $(\pi_A, j, \pi_B, \pi_b, \pi_{b_2}, \ldots, \pi_{b_h}, \pi_{B_{b+1}}, q, \pi_d, \pi_E)$. We use $\pi_{B_1}$ and $\pi_{b_i}$ to distinguish between subsets of the sets $\{J_{j+1}, \ldots, J_n\}$ and $\{J_1, \ldots, J_{j-1}\}$; the sets $\pi_{B_1}$ and $\pi_{B_{b+1}}$ can be empty. Define $\mu_0$ as the sequence of the jobs $J_1, \ldots, J_j$ obtained by removing the jobs $J_{j+1}, \ldots, J_n$ from $s^*$; $\mu_0 = (\pi_a, j, \pi_b, \ldots, \pi_{b_h}, q, \pi_d, \pi_e)$. Define $\mu_i$ $(i = 1, \ldots, h)$ as the sequence of the jobs $J_1, \ldots, J_j$ obtained by inserting $J_j$ in $\sigma_{j-1}$ between $\pi_{b_i}$ and $\pi_{b_{i+1}}$, that is,

$$
\mu_i = (\pi_a, \pi_{b_1}, \ldots, \pi_{b_i}, j, \pi_{b_{i+1}}, \ldots, \pi_{b_h}, q, \pi_d, \pi_e), \quad \forall \ i = 1, \ldots, h.
$$

Note that all sequences $\mu_i$ $(i = 0, \ldots, h)$ are obtained by inserting $J_j$ in $\sigma_{j-1}$; hence, $f(\sigma_j) - f(\mu_i) \leq 0$ for $i = 0, \ldots, h$, where $\sigma_j$ has the form $(\pi_a, \pi_b, \ldots, \pi_{b_h}, q, \pi_d, j, \pi_e)$. Let $s$ be the schedule obtained from $s^*$ by shifting $J_j$ and the jobs in $\pi_{B_i}$ $(i = 1, \ldots, h + 1)$ to immediately after the last job in $\pi_d$, that is,

$$
s = (\pi_A, \pi_{b_1}, \ldots, \pi_{b_h}, q, \pi_d, j, \pi_{B_1}, \ldots, \pi_{B_{b+1}}, \pi_E).
$$

We will show that the cost of $s$ is no more than the cost of $s^*$, that $s$ satisfies Lemmas 2 and 3, and that in $s$ all equal jobs are executed in order of increasing index.

To show $f(s) - f(s^*) \leq 0$, we first need to compute $f(\sigma_j) - f(\mu_i)$, for $i = 0, \ldots, h$.

We start with revealing the situation concerning the new-max jobs. In $s^*$, all jobs in $\pi_{b_g}$ $(g = 1, \ldots, h)$ succeed $J_j$, whereas all these jobs have index smaller than $j$. As $s^*$ satisfies Lemma 2 and all equal jobs are executed in order of increasing index in $s^*$, all jobs in $\pi_{b_g}$ $(g = 1, \ldots, h)$ must have processing time larger than $p_j$. Hence, all jobs in $\pi_{b_g}$ $(g = 1, \ldots, h)$ face the same new-max job in $s$ as in $s^*$. Job $j$ and the jobs in $\pi_{B_g}$ $(g = 1, \ldots, h + 1)$ have $J_q$ as its new-max job in $s$, as $J_q$ is a new-max job in $\sigma_j$ and $s^*$ satisfies Lemma 2. Let $\tilde{p}_i$ denote the processing time of the new-max job faced by $J_j$ in $\mu_i$. Taking all this into account, we obtain that

$$
\begin{align*}
f(\sigma_j) - f(\mu_i) & = w_j \left[ \sum_{g=i+1}^{h} p(\pi_{b_g}) + p_q + p(\pi_d) + (m-1)(p_q - \tilde{p}_i) \right] - \\
p_j \left[ \sum_{g=i+1}^{h} w(\pi_{b_g}) + w_q + w(\pi_d) \right], & \quad \forall \ i = 0, \ldots, h.
\end{align*}
$$

Since $s^*$ satisfies Lemma 2, the new-max job faced by $J_j$ in $s^*$ is a job in $\pi_a$ or $J_j$ is a new-max job itself. Hence, the new-max job faced by $J_j$ in $s^*$ is the same one $J_j$ faces in $\mu_0$; this job has processing time $\tilde{p}_0$. Similarly, the jobs in $\pi_{B_i}$ $(i = 1, \ldots, h + 1)$ have, either the same new-max job in $s^*$ as $J_j$ faces in $\mu_{i-1}$, or a job with larger processing time.
from among the jobs in $\pi_{B_i}$; hence, the processing time of the new-max job faced by the jobs in $\pi_{B_i}$ is at least equal to $\tilde{p}_{i-1}$. Taking this into account, we obtain

$$f(s) - f(s^*) \leq$$

$$w_j \left[ \sum_{g=1}^{h} p(\pi_{b_g}) + p_q + p(\pi_d) + (m - 1)(p_q - \tilde{p}_0) \right] - p_j \left[ \sum_{g=1}^{h} w(\pi_{b_g}) + w_q + w(\pi_d) \right] +$$

$$w(\pi_{B_i}) \left[ \sum_{g=1}^{h} p(\pi_{b_g}) + p_q + p(\pi_d) + (m - 1)(p_q - \tilde{p}_0) \right] - p(\pi_{B_i}) \left[ \sum_{g=1}^{h} w(\pi_{b_g}) + w_q + w(\pi_d) \right] + \ldots +$$

$$w(\pi_{B_{h+1}}) \left[ p_q + p(\pi_d) + (m - 1)(p_q - \tilde{p}_0) \right] - p(\pi_{B_{h+1}}) \left[ w_q + w(\pi_d) \right] \equiv$$

$$f(\sigma_j) - f(\mu_0) + \Delta_1 + \ldots + \Delta_{h+1},$$

where $\Delta_i$ ($i = 1, \ldots, h + 1$) is the part concerning $\pi_{B_i}$. As we want to show that $f(s) - f(s^*) \leq 0$, it is sufficient to prove that $\Delta_i \leq 0$, as we already know that $f(\sigma_j) - f(\mu_0) \leq 0$.

To show that $\Delta_i \leq 0$, we prove that $w(\pi_{B_i}) \left[ f(\sigma_j) - f(\mu_{i-1}) \right] - w_j \Delta_i \geq 0$ for $i = 1, \ldots, h+1$.

We have

$$w(\pi_{B_i}) \left[ f(\sigma_j) - f(\mu_{i-1}) \right] - w_j \Delta_i =$$

$$w(\pi_{B_i}) \left[ w_j \sum_{r=1}^{h} p(\pi_{b_r}) + p_q + p(\pi_d) + (m - 1)(p_q - \tilde{p}_{i-1}) \right] -$$

$$p_j \left[ \sum_{r=1}^{h} w(\pi_{b_r}) + w_q + w(\pi_d) \right] -$$

$$w_j \left[ w(\pi_{B_i}) \sum_{r=1}^{h} p(\pi_{b_r}) + p_q + p(\pi_d) + (m - 1)(p_q - \tilde{p}_{i-1}) \right] +$$

$$p(\pi_{B_i}) \left[ \sum_{r=1}^{h} w(\pi_{b_r}) + w_q + w(\pi_d) \right] =$$

$$\left[ \sum_{r=1}^{h} w(\pi_{b_r}) + w_q + w(\pi_d) \right] \left[ w_j p(\pi_{B_i}) - p_j w(\pi_{B_i}) \right] \geq 0,$$

since each job $k$ in $\pi_{B_i}$ has $w_k / p_k \leq w_j / p_j$.  

11
If \( \Delta_i < 0 \) for some \( i \in \{1, \ldots, h+1\} \), then we obtain \( f(s) - f(s^*) < 0 \), which contradicts the supposed optimality of \( s^* \). Hence, we are done unless all \( J_k \) in \( \{ \pi_{B_k} \cup \ldots \cup \pi_{B_{k+1}} \} \) have \( w_k/p_k = w_j/p_j \). Suppose this is the case; as \( s^* \) satisfies Lemma 2, we therefore must have \( p_k \geq p_j \) for all \( J_k \) in \( \{ \pi_{B_k} \cup \ldots \cup \pi_{B_{k+1}} \} \). From this observation it follows that \( s \) satisfies Lemmas 2 and 3 and that all equal jobs are processed in order of nondecreasing index in \( s \). This settles the proof for the case that \( J_j \) is scheduled before \( J_q \) in \( s^* \).

**Proof of (iii).** In a similar fashion, we deal with the case that in \( s^* \) \( J_j \) is scheduled not only after \( J_q \) but after some other jobs from \( \{ J_1, \ldots, J_{j-1} \} \) that \( J_j \) is not supposed to succeed as well. Let \( s^* \) have the form \((\pi_A, q, \pi_d, \pi_E, j, \pi_G)\), where \((q, \pi_d, j)\) is the segment belonging to \( J_q \) in \( \sigma_j \) (the proof that \( J_q \) is a new-max job in \( s^* \) and that the first part of the segment belonging to \( J_q \) has this form is similar to the proof given in the analysis of the previous case). Let \( \sigma^* \) be the sequence of the jobs \( J_1, \ldots, J_j \) obtained by removing the other jobs from \( s^* \); \( \sigma^* = (\pi_a, q, \pi_d, \pi_e, j, \pi_g) \). We have that \( \sigma_j = (\pi_A, q, \pi_d, j, \pi_e, \pi_g) \), and we will show that \( s = (\pi_A, q, \pi_d, j, \pi_E, \pi_G) \) is a schedule with the desired properties. Before starting our computations, we look at the situation concerning the new-max jobs in \( s^* \) compared to \( s \) and \( \sigma^* \) compared to \( \sigma_j \). Since \( p_j \leq p_q \) and \( J_j \) succeeds \( J_q \) in each schedule under consideration, nothing changes with respect to the new-max status of any job except for \( J_j \). Since \( s^* \) satisfies Lemma 2, the new-max job faced by \( J_j \) in \( s^* \), say \( J_a \), is a job from the set \( \{ J_1, \ldots, J_{j-1} \} \); hence, \( J_j \) has \( J_a \) as new-max job in both \( s^* \) and \( \sigma^* \).

In both \( \sigma_j \) and \( s \), \( J_j \) has \( J_q \) as new-max job. Taking this into account, we obtain

\[
f(\sigma^*) - f(\sigma_j) = w_j[p(\pi_e) + (m-1)(p_a - p_q)] - p_j w(\pi_e).
\]

Since both sequences can be obtained by inserting \( J_j \) in \( \sigma_{j-1} \) and \( \sigma_j \) is selected by Algorithm WSPT-MCI, whereas \( J_j \) is inserted into an earlier spot in \( \sigma_j \) than in \( \sigma^* \), we have \( f(\sigma^*) - f(\sigma_j) > 0 \). We further have

\[
f(s^*) - f(s) = w_j[p(\pi_E) + (m-1)(p_a - p_q)] - p_j w(\pi_E) =
\]

\[
f(\sigma^*) + w_j p(\pi_E \setminus \pi_e) - p_j w(\pi_E \setminus \pi_e) > 0,
\]

since \( f(\sigma^*) - f(\sigma_j) > 0 \) and \( w_j/p_j \geq w_i/p_i \) for each job \( i \in (\pi(E) \setminus \pi(e)) \). This clearly contradicts the supposed optimality of \( s^* \).

On basis of induction, we know that Algorithm WSPT-MCI determines an optimal schedule. \( \square \)

## 5 Conclusions

We have presented an algorithm that solves the problem under consideration in \( O(n^2) \) time for an arbitrary number of machines. In case of a single machine, the problem is solved through the WSPT rule, which runs in \( O(n \log n) \) time. When the number of
machines is very large, then the optimal rule is SPT, which is again very easy. (In this extreme case the weights do not have any effect on the optimal schedule.) One interesting question concerns the determination of a lower bound on the number of machines such that the SPT rule is guaranteed to yield an optimal solution. A far more interesting problem, however, is what we can do for the special case of the flow shop problem with processing times $p_{ij} = p_j/s_i$.

Acknowledgements

Part of this research was conducted when the first author visited Eindhoven University of Technology, which visit was made possible by a grant from Nuffic. The first and second author were supported by a grant from INTAS project nr. 93-257.

References


