J Math Model Algor (2007) 6:563–576
DOI 10.1007/s10852-007-9057-6

Total Completion Time in a Two-machine Flowshop with Deteriorating Tasks

G. Finke · A. Oulamara

Received: 3 April 2006 / Accepted: 10 January 2007 /
Published online: 6 March 2007

Abstract This paper deals with two-machine flowshop problems with deteriorating tasks, i.e. tasks whose processing times are a nondecreasing function that depend on the length of the waiting periods. We consider the so-called *Restricted Problem*. This problem can be defined as follows: for a given permutation of tasks, find an optimal placement on two machines so that the total completion time is minimized. We will show that the *Restricted Problem* is nontrivial. We give some properties for the optimal placement and we propose an optimal placement algorithm.

Keywords Flowshop · State-dependent processing time · Total completion time.

Mathematics Subject Classifications (2000) 90B30 · 90B35 · 90C05

1 Introduction

In this paper, we consider the two-machine flowshop problem with *deteriorating tasks*. Such a problem consists of two machines that are continuously available and a set of tasks to be processed. Each task has two operations to be sequentially processed on the two machines. The tasks have deteriorating processing times, i.e. the processing time on the second machine is a continuous nondecreasing function of the waiting time between machines. Such a deterioration appears, for instance,

G. Finke
Laboratoire Leibniz-IMAG, 46 Av. Felix Viallet, 38031 Grenoble, France

A. Oulamara (✉)
MACSI Project LORIA – INRIA Lorraine, Ecole des Mines de Nancy,
Parc de Saurupt, 54042 Nancy, France
e-mail: Oulamara@loria.fr

in the steel production where the material will cool during the waiting periods and has to be reheated for the subsequent process. A similar situation will also occur in scheduling maintenance tasks, where the maintenance time depends on the length of time elapsed since the last maintenance operation.

Two kinds of deteriorating tasks exist in the literature. The first focuses on resource dependent processing. There the processing times are functions of resources assigned to its processing, Janiak [6, 7]. In the second case, the processing times are nondecreasing functions of the time they are released into the system, Sriskandarajah and Goyal [11], Wagneur and Sriskandarajah [13, 14] and Mosheiov [10].

Several articles study scheduling problems with deterioration of the processing times. Most of these studies treat single machine problems, see e.g. Alidaee [1], Gupta and Gupta [5] and Mosheiov [8]. There the processing times are linear or nonlinear nondecreasing functions in which the processing times increase with time elapsed since the release into the system. For parallel machines problems, Chen [2], Mosheiov [9] consider deteriorating tasks, they assume a linear deterioration and study the minimum completion time and makespan criteria.

For classical flowshop scheduling problems, Sriskandarajah and Goyal[11], Wagneur and Sriskandarajah [13, 14], and Finke and Jiang [3] studied various forms of deteriorating tasks in which the processing times of tasks are state-dependent on the time that the tasks spend in the system. Sriskandarajah and Wagneur [12] report complexity results of the two-machine flowshop problem for the makespan criterion, and they propose in [13] a control vector for the start times of the tasks for the makespan, lateness and tardiness criteria if the order of the tasks is given. Finke and Jiang [3] and Finke et al. [4] consider the equivalent reverse model, in which the deteriorating processing time of tasks is given on the first machine. For a given order of tasks, they propose a greedy placement algorithm for the makespan, lateness and tardiness criteria and they generalize their algorithm to the m-machine flowshop problem.

In this paper, we consider the restricted problem in which for a given sequence of tasks, one wants to find an optimal placement that minimizes the total completion time. This problem is trivial for classical two-machine flowshop problem. It is sufficient to schedule tasks as soon as possible. However finding an optimal sequence of tasks in order to minimize the total completion time is strongly NP-hard. The aim here is to suggest an optimal algorithm to solve the restricted problem. Section 2 is dedicated to some definitions and notations used in this article. In Section 3 we define a shift operation within a certain block structure, and in Section 4, we give some properties and the optimal placement algorithm of tasks. Finally in Section 5, we consider the linear deterioration case, which is then compared to the linear programming approach.

2 Definitions and Notations

A set of tasks $N = \{1, 2, \ldots, n\}$ is given and every task T_j consists of two operations $O_{1,j}$ and $O_{2,j}$, $j \in N$. These operations are to be processed in a two-machine flowshop, i.e. operation $O_{1,j}$ must be completed on the upstream machine M_1 before operation $O_{2,j}$ can start on the downstream machine M_2. The processing time of task T_j is state-dependent. The operation $O_{1,j}$ has processing time $p_{1,j}$ on machine M_1. The processing time of $O_{2,j}$ depends on its starting time on M_2, i.e. if σ_j is the time

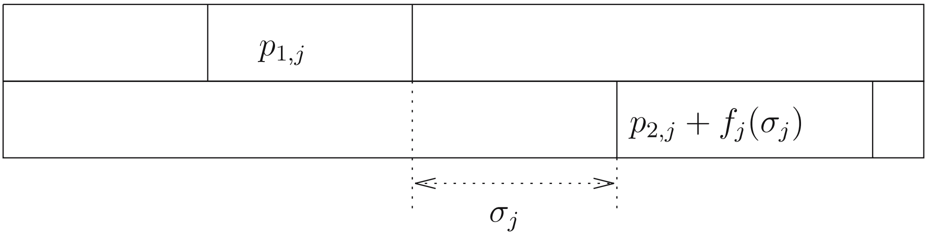

Fig. 1 Processing time of task T_j

difference between the end of $O_{1,j}$ and start of $O_{2,j}$, then the processing time of $O_{2,j}$ is $F_j(\sigma_j) = p_{2,j} + f_j(\sigma_j)$, where f_j is a nondecreasing function of σ_j, $\sigma_j \geq 0$ (Fig. 1). We assume that $f_j(0) = 0$.

We are interested here in the placement of the tasks that minimizes the total completion time $\sum C_j$, where C_j indicates the completion time of task T_j. This problem is called *Restricted Problem* which can be defined as follows:

Restricted Problem For a given permutation π of tasks, find an optimal placement of these tasks on both machines so that the total completion time is minimized (Fig. 2). We give an example below to illustrate this Restricted Problem.

Example 1 Consider a set of three tasks with their processing times as shown below (Table 1). Assume that the deterioration of each task is given by a function $F_j(\sigma_j) = p_{2,j} + f_j(\sigma_j)$, where $f_1(\sigma_j) = f_3(\sigma_j) = \sigma_j$ and $f_2(\sigma_j) = \frac{1}{4}\sigma_j^2$.

It appears that the Restricted Problem is nontrivial (Fig. 4). For the placement of each task, one has to determine the best compromise between the earliest release time on the first machine (Fig. 3) and the latest which is the no-wait problem (Fig. 2). Finke and Jiang [3] made the same remarks for the makespan criterion.

Definition 1 Denote by π_{nw} the no-wait placement of tasks $T_1, \ldots, T_n$. We define a block B of tasks as succession of tasks $T_i, T_{i+1}, \ldots, T_j$, such that there is no idle-time between these tasks on the second machine and B is maximal with this property.

From this definition, we have the following property.

Property 1 *A no-wait placement π_{nw} of tasks $T_1, T_2, \ldots, T_n$ is a succession of blocks* (Fig. 5).

Table 1 Tasks processing times

	T_1	T_2	T_3
$p_{1,j}$	1	1	4
$p_{2,j}$	5	1	1

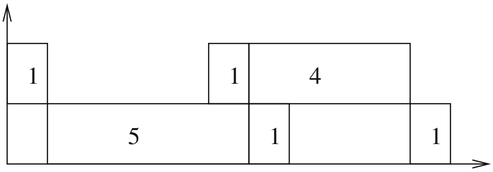

Fig. 2 No-wait placement, $\sum C_i = 24$

3 Shift Operation in Blocks

Let $B_1, \ldots, B_l$ be the block partition of tasks $T_1, \ldots, T_n$ in the no-wait placement π_{nw}. Denote by $n_1, \ldots, n_l$ the number of tasks in blocks $B_1, \ldots, B_l$ respectively, with $\sum_{i=1}^{l} n_i = n$. For each block B_k denote by $\Delta_{n_k-1}(k)$, $\Delta_{n_k-2}(k), \ldots, \Delta_1(k)$ the idle time intervals between tasks of block B_k on first machine and denote by γ_k $(k = 1, \ldots, l-1)$ the idle time interval between B_k and B_{k+1} on the second machine (Fig. 6).

Starting from placement π_{nw}, a reduction of the total completion time can be obtained by a reduction of the idle-time γ_k $k = 1, \ldots, l-1$ on the second machine. However, the reduction of γ_k requires a shifting of the block B_{k+1} to the left in the current placement π_{nw}. This operation should be possible if the tasks sequenced after the last idle time interval of the block B_k can also be shifted to the left. Thus we define a left-shift operation in a block as follows.

Definition 2 A left-shift operation in block B_k applied to the idle time interval $\Delta_1(k)$ consists of a shift to the left of all tasks sequenced after the interval $\Delta_1(k)$ with a value ϵ, such that $\epsilon \leq \Delta_1(k)$.

The left-shift operation on $\Delta_1(k)$ modifies a placement π_{nw} as follows:

1. The placement of blocks $B_1, \ldots, B_{k-1}$ is unchanged.
2. The placement of the tasks sequenced before the idle time interval $\Delta_1(k)$ in block B_k is unchanged.
3. All tasks sequenced after $\Delta_1(k)$ should be shifted to the left by ϵ.
4. Update the block structure (after step 3, one may shift the blocks $B_{k+1}, \ldots, B_l$ to the left, leading possibly to the fusion of blocks).

The shifted tasks in block B_k induce a deterioration of the processing times of these tasks on the second machine. Thus, the completion time of block B_k is increased. Denote this increase α_k.

We are interested here to the shift value ϵ for which α_k never exceeds the value of γ_k (idle time between B_k and B_{k+1}) on the second machine.

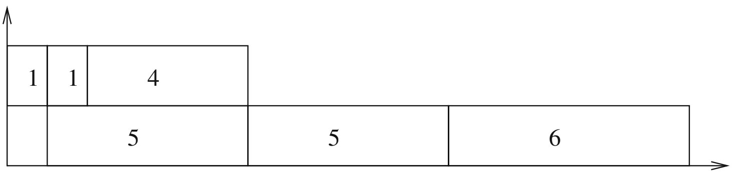

Fig. 3 Earliest placement, $\sum C_i = 34$

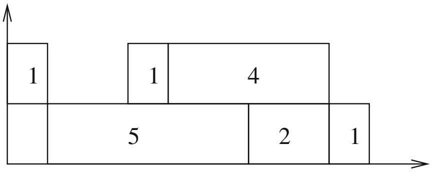

Fig. 4 Optimal placement, $\sum C_i = 23$

Definition 3 A block B_k is called "improvable" if the left-shift operation on the last idle time interval $\Delta_1(k)$ of block B_k reduces the total completion time of the current placement.

Definition 4 A block B_k is called "stable" if one of the following conditions holds:

1. The left-shift operation applied to the last idle time interval $\Delta_1(k)$ of the block B_k increases the total completion time of the current placement.
2. There is no idle time on the first machine between the tasks of block B_k.

From these definitions, we deduce that any nonimprovable block is stable and any block made up of one task is also stable.

Definition 5 An improvable block B_k is an absorbant block, if the left-shift operation on the last idle time interval $\Delta_1(k)$ of block B_k, leads to the disappearance of the idle time interval γ_k.

Whenever a block becomes absorbant, all blocks sequenced after this one should be reindexed.

4 Optimal Placement

In this part, we show that in an optimal placement, all idle time intervals on both machines are essential, i.e. any modification of these idle times involves an increase of the total completion time.

Let $T_i(j)$ be the i^{th} task of block B_j and denote by $\sigma_i(j)$ the waiting-time between the completion time of the task $T_i(j)$ on the first machine and its starting time on the second machine. Assume that $\Delta_i(j)$ is the idle time on the first machine between $T_i(j)$ and $T_{i+1}(j)$.

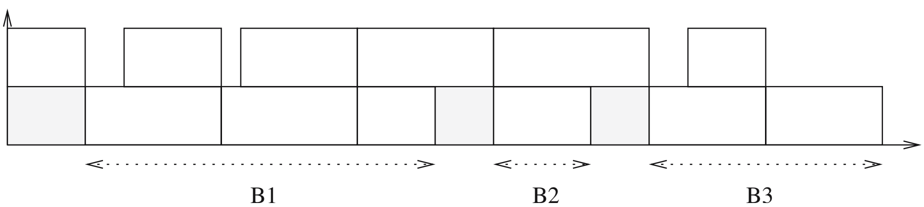

Fig. 5 Decomposition into blocks

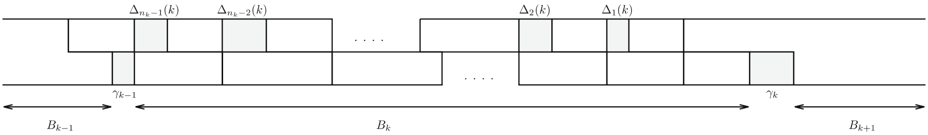

Fig. 6 Decomposition into blocks

Property 2 *For each block B_j in an optimal placement, we have*

$$\forall\ T_i(j) \in B_j,\quad 1 \le j \le l\ :\ \min\{\sigma_i(j), \Delta_i(j)\} = 0$$

Proof Assume that there is an optimal placement π such that for a given task $T_i(k)$ in block B_k, $\min\{\sigma_i(k); \Delta_i(k)\} \neq 0$ (Fig. 7). Let π' be a placement obtained from π by shifting to the right the task $T_i(k)$ by ϵ on the first machine, where

$$\epsilon = \begin{cases} \sigma_i(k) & \text{if } \ \Delta_i(k) \ge \sigma_i(k) \\ \\ \Delta_i(k) & \text{if } \ \Delta_i(k) < \sigma_i(k) \end{cases}$$

The processing time of $T_i(k)$ on the second machine is decreased in π' by $f_i(\sigma_i(k))$, thus $\sum_{i=1}^{n} C_{2,i}(\pi') < \sum_{i=1}^{n} C_{2,i}(\pi)$, which contradicts the fact that π is an optimal placement. □

This property shows that in an optimal placement, each task followed by an idle time on the first machine is sequenced in a no-wait fashion.

Property 3 *Let B_k be a block in a given placement π and $T_s(k), T_{s+1}(k), \ldots, T_{n_k}(k)$ the tasks sequenced after the last idle-time interval $\Delta_{s-1}(k)$ on the first machine. If B_k is stable by applying the left-shift operation on the last idle time interval $\Delta_{s-1}(k)$, then B_k is also stable by applying any left-shift operation on the intervals $\Delta_{n_k-1}(k)$, $\Delta_{n_k-2}(k), \ldots, \Delta_{s-2}(k)$.*

Proof Consequence of Property 2 and the definition of stable block. □

Lemma 1 *An optimal placement of tasks is a succession of stable blocks.*

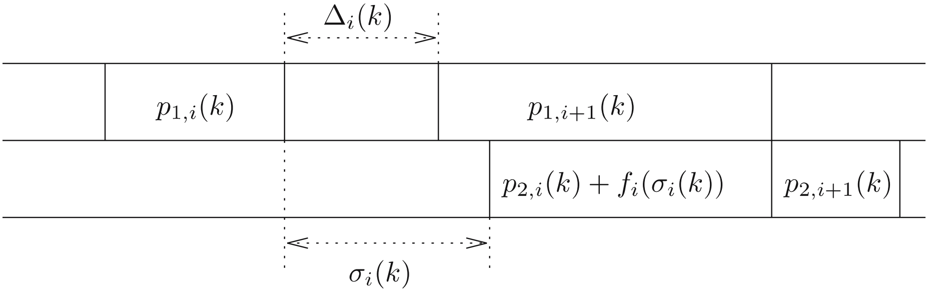

Fig. 7 Processing time of task T_j

Proof Let π be an optimal placement of tasks. Let $B_1, \ldots, B_n$ be a block partition of π. Assume that B_k is not stable. Applying the left-shift operation on the last idle time interval of the block B_k improves the value of the total completion time, which contradicts the fact that π is an optimal placement. □

Given a block partition of a no-wait placement, the value of the total completion time can be improved by a reduction of the idle time intervals on the second machine. Indeed, at step i of the algorithm, we test if the block B_i is improvable by applying the left-shift operation on the last idle time interval of B_i. If B_i is improvable, we calculate the greatest value ϵ for which the total completion time is improved. Thus we update the new placement and we repeat the same operation on the last idle time interval of the block B_i. Otherwise, if B_i is not improvable, the algorithm goes to the next block. The algorithm can be described as follows:

Placement Algorithm

1. Let π_{nw} be the no-wait placement of the tasks and $B_1, \ldots, B_l$ the block partition of π, $k = 1$.
2. If B_k is an improvable block go to 3, else go to 4.
3. Apply the left-shift operation of value ϵ to the last idle time interval on the first machine of the block B_k. We obtain two cases:

   3a. The last idle time interval of B_k vanishes. Then repeat 3.
   3b. The idle time interval between B_k and B_{k+1} on the second machine disappears. Then B_k is an absorbant block. B_k and B_{k+1} yield the same block and the blocks which follow are reindexed, repeat 3.

4. If $k = l$ stop, else $k = k + 1$, go to 3.

Theorem 1 *The placement obtained by the previous algorithm is optimal for the total completion time criterion.*

Proof Let $B_1, \ldots, B_r$ denote the block partition of a placement π obtained by the algorithm. We show that there is an optimal solution π' with blocks $B'_1, \ldots, B'_s$ for which $r = s$ and $B_1 = B'_1$. Since the reduced sequence $\pi - \{\text{tasks of } B_1\}$ yields the blocks $B_2, \ldots, B_r$, we can repeat the argument (induction on the number of blocks) and show that $r = s$ and $B_i = B'_i$ for all i.

In order to obtain an optimal solution π' with $B_1 = B'_1$, let us consider the optimal solution π' for which $|n_1 - n'_1|$ is minimal, where n_1 and n'_1 is the number of tasks in the block B_1 and B'_1, respectively. According to the Lemma 3, the block B'_1 is stable. We distinguish two cases:

Case 1 If $n'_1 < n_1$. From blocks B_1 and B'_1, we built the block B''_1 and B''_2 as follows:

– $B''_1 = B'_1$,
– Schedule the rest of the tasks exactly as they are placed in B_1. Denote B''_2 the block built by this operation.

We know that B_1 is obtained by applying a successive left-shift operation on the idle time intervals of the improvable blocks, in nondecreasing order of their indices. Moreover, the idle time between B''_1 and B''_2 on the second machine does not exist in B_1, therefore B''_1 is an improvable block. However, $B''_1 = B'_1$, we have a contradiction to the fact that B'_1 is a stable block, which implies that n'_1 cannot be strictly less than n_1.

Case 2 If $n'_1 > n_1$. From blocks B_1 and B'_1, we built the block B''_1 and B''_2 as follows:

- $B''_1 = B_1$,
- Schedule the rest of the tasks exactly as they are placed in B'_1, Denote B''_2 the block built by this placement.

B'_1 is divided into two blocks B''_1 and B''_2 in the new placement, with γ''_1 the idle time between B''_1 and B''_2 on the second machine. However γ''_1 does not exist in B'_1 so that this idle time interval is removed by applying a left-shift operation on B''_1.

According to the placement π, B_1 is a stable block and $B_1 = B''_1$, which implies that any application of the left-shift operation on B''_1 increases the value of the total completion time. This contradicts the fact that π' is an optimal placement. According to Cases 1 and 2, $n'_1 = n_1$. B_1 and B'_1 are composed of the same set of tasks and are stable. The placement of the tasks in B'_1 is exactly the same in B_1, i.e. the placement obtained by the previous algorithm is optimal for the total completion time criterion.

□

5 Linear Deterioration Case

We consider here the case in which the deterioration of tasks are given by a linear function $F_j(\sigma_j) = p_{2,j} + f_j \times \sigma_j$. We describe the optimal placement and how to obtain it in order to minimize the total completion time. We shall adapt the algorithm to this case and show that in Theorem 2 that its complexity is $O(n^2)$. For the linear case, one has also as an alternate solution method a linear programming approach. Let us first formulate our placement problem in form of a linear program and then compare the complexities.

Let $C_{1,j}$, $C_{2,j}$ be the completion time of task j on machine 1 and 2, respectively. We get the following linear program:

$$
\begin{aligned}
&\min \sum_{j=1}^{n} C_{2,j} \\
&C_{1,j} - p_{1,j} \geq C_{1,j-1} \quad j = 2, \ldots, n \quad (1) \\
&C_{2,j} - p_{2,j} \geq C_{2,j-1} + f_j(C_{2,j-1} - C_{1,j}) \quad j = 2, \ldots, n \quad (2) \\
&C_{2,j} - p_{2,j} \geq C_{1,j} \quad j = 1, \ldots, n \quad (3) \\
&C_{1,1} = p_{1,1}, \ C_{2,2} = p_{1,1} + p_{2,1}, \\
&C_{1,j} \geq 0, \ C_{2,j} \geq 0, \quad j = 2, \ldots, n
\end{aligned}
$$

Constraints (1) describe the succession of operations on M_1. From the flowshop precedence constraints $C_{1,j} + \sigma_j + f_j\sigma_j + p_{2,j} = C_{2,j}$ and $\sigma_j \geq 0$, one obtains $\sigma_j = \frac{C_{2,j} - C_{1,j} - p_{2,j}}{1+f_j}$ and constraints (3). Replacing σ_j in the constraints expressing the succession of operations on M_2, i.e. $C_{2,j} - p_{2,j} - f_j\sigma_j \geq C_{1,j-1}$, yields the the constraints (2) of the given linear program.

The experience with simplex based solutions gives a practical complexity of $O(N \times C^2)$, where N is the number of variables and C the number of constraints. In the given program, $N = 2n - 2$ and $C = 3n$. Hence, we get the complexity $O(n^3)$, which is worse than the $O(n^2)$ complexity for our method.

Now we give properties of the optimal placement that will be used for the adapted algorithm.

Lemma 2 *Let B_k be a block in given placement π and $T_s(k)$, $T_{s+1}(k)$, …, $T_{n_k}(k)$ the tasks sequenced after the last idle-time interval $\Delta_{s-1}(k)$ on the first machine. B_k is being improved by the left-shift operation on $\Delta_{s-1}(k)$ if*

$$\sum_{i=s}^{n_k}(n_k+1-i)\times f_i(k)\times\beta_i(k)-\sum_{i=k+1}^{l}n_j\leq 0,$$

where $\beta_s(k)=1$ and $\beta_i(k)=\prod_{t=s}^{i-1}(1+f_t(k))$, $s+1\leq i\leq n_k$.

Proof Let π' be the placement obtained by applying the left-shift operation on $\Delta_{s-1}(k)$ with value ϵ. The completion times of tasks in π' are:

1. $\forall i,\ T_i(j)\in B_j, 1\leq i\leq n_j, 1\leq j\leq k-1 : C'_{2,i}(j)=C_{2,i}(j)$
2. $\forall i,\ T_i(k)\in B_k, 1\leq i\leq s-1 : C'_{2,i}(k)=C_{2,i}(k)$
3. $\forall i,\ T_i(k)\in B_k, s\leq i\leq n_k-1$ we have

$$\begin{aligned}
C'_{2,s}(k) &= C_{2,s}(k)+f_s(k)\times\epsilon\\
C'_{2,s+1}(k) &= C_{2,s+1}(k)+f_s(k)\times\epsilon+f_{s+1}(k)(f_s(k)\times\epsilon)+f_{s+1}(k)\times\epsilon\\
&= C_{2,s+1}(k)+f_s(k)\times\epsilon+f_{s+1}(k)(1+f_s(k))\times\epsilon\\
\ldots &\quad \ldots\\
C'_{2,i}(k) &= C_{2,i}(k)+f_s(k)\times\epsilon+f_{s+1}(k)\times(1+f_s(k))\times\epsilon+\ldots\\
&\quad +f_i(k)(1+f_{i-1}(k))\times\ldots\times(1+f_s(k))\times\epsilon.
\end{aligned}$$

A block B_k is improvable if $\displaystyle\sum_{i=1}^{n}C_{2,i}(\pi')-\sum_{i=1}^{n}C_{2,i}(\pi)\leq 0$

$$\begin{aligned}
\sum_{i=1}^{n}C'_{2,i}(\pi)-\sum_{i=1}^{n}C_{2,i}(\pi') &= \sum_{j=1}^{l}\sum_{i=1}^{n_j}C_{2,i}(j)+\epsilon\times\sum_{i=s}^{n_k}(n_k+1-i)\,f_i(k)\times\beta_i(k)\\
&\quad -\epsilon\sum_{j=k+1}^{l}n_j-\sum_{j=1}^{l}\sum_{i=1}^{n_j}C_{2,i}(j)
\end{aligned}$$

with $\beta_s(k)=1$ and $\beta_i(k)=\displaystyle\prod_{t=s}^{i-1}(1+f_t(k)), s+1\leq i\leq n_k$ Thus, B_k is improvable if

$$\sum_{i=s}^{n_k}(n_k+1-i)\,f_i(k)\times\beta_i(k)-\sum_{i=k+1}^{l}n_j\leq 0$$ □

Lemma 3 *If B_k is improvable by the left-shift operation on the last idle time interval $\Delta_{s-1}(k)$ of the block B_k, then the greatest value of ϵ is:*

$$\epsilon=\min\left\{p_{2,s-1}(k)-p_{1,s}(k)-\sigma_s(k)\ ;\ \frac{\gamma_k}{1+\sum_{i=s}^{n_k}f_i(k)\times\beta_i(k)}\right\}$$

Proof Let B_k an improvable block. It is obvious that the value of ϵ for which the left-shift operation is applied on the last idle time interval $\Delta_{s-1}(k)$ of B_k does not exceed the value of $\Delta_{s-1}(k)$. In other words, $\epsilon \leq \Delta_{s-1}(k)$, therefore the first upper bound of ϵ is $\epsilon_1 = \Delta_{s-1}(k)$. From the Property 2, it easy to check that $\Delta_{s-1}(k) = p_{2,s-1} - p_{1,s} - \sigma_s(k)$.

Also, according to the definition of the left-shift operation, ϵ does not exceed the value ϵ_2 for which the idle time interval γ_k between B_k and B_{k+1} is removed. In other words, ϵ_2 is a solution of the equation:

$$C'_{1,1}(k+1) - C'_{2,n_k}(k) = 0$$

$$\Longrightarrow (C_{1,1}(k+1) - \epsilon_2) - \left(C_{2,n_k} + \epsilon \sum_{i=s}^{n_k} f_i(k) \times \beta_i(k) \right) = 0,$$

with $\beta_s(k) = 1$ and $\beta_i(k) = \prod_{t=s}^{i-1}(1 + f_i(k)), s+1 \leq i \leq n_k$
Thus,

$$C_{1,1}(k+1) - C_{2,n_k}(k) = \left(1 + \sum_{i=s}^{n_k} f_i(k) \times \beta_i(k) \right) \epsilon$$

However,

$$C_{1,1}(k+1) - C_{2,n_k}(k) = \gamma_k \quad \Rightarrow \quad \epsilon_2 = \frac{\gamma_k}{1 + \sum_{i=s}^{n_k} f_i \times \beta_i(k)}$$

We have $\epsilon \leq \{\epsilon_1, \epsilon_2\}$, thus the greatest value of ϵ is $\epsilon = \min\{\epsilon_1, \epsilon_2\}$ □

Remark 1 If $\epsilon = \epsilon_1$, the last idle time interval $\Delta_{s-1}(k)$ of B_k is removed in the new placement. If $\epsilon = \epsilon_2$, then there is no idle time between B_k and B_{k+1}. In the new placement on the second machine, in this case, B_k and B_{k+1} are combined and form the same block.

Theorem 2 *For the linear case, the Placement Algorithm has complexity $O(n^2)$.*

Proof For a given last idle-time interval $\Delta_s(k)$ on the first machine of block B_k, testing if B_k is being improved by the left-shift operation on $\Delta_s(k)$ is in $O(n)$ (i.e. calculating $\beta_i(k)$ and checking if $\sum_{i=s}^{n_k}(n_k + 1 - i) \times f_i(k) \times \beta_i(k) - \sum_{i=k+1}^{l} n_j \leq 0$). Since there are at most $n - 1$ idle-time intervals on the first machine, we get the overall complexity $O(n^2)$. □

Example Consider a set of seven tasks with their processing times and deterioration rate as shown in Table 2.

Table 2 Tasks processing times and deterioration rate

	T_1	T_2	T_3	T_4	T_5	T_6	T_7
$p_{1,j}$	1	1	1	3	1	8	2
$p_{2,j}$	3	2	2	2	1	3	1
f_j	1	0.5	0.25	0.25	0	0.5	0.5

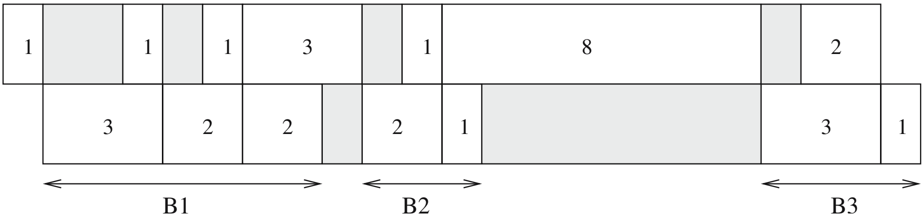

Fig. 8 No-wait Placement, $\sum C_i = 86$

Algorithm Given a no-wait placement with $\sum C_i = 86$ and the block partition B_1, B_2, B_3 (Fig. 8)

- $k = 1$
  - Test if B_1 can be improved by applying a left-shift operation to the last idle time

    $f_3(1) - (n_2 + n_3) = 0.25 - (2 + 2) = -3.5 < 0$

    $\Rightarrow B_1$ is an improvable block.
  - Calculate the value of ϵ.

    $\epsilon_1 = p_{2,2} - p_{1,3} - \sigma_2(2) = 2 - 1 - 0 = 1$
    $\epsilon_2 = \frac{\gamma_1(1)}{1+f_3(1)} = \frac{3-2}{1+0.25} = 0.8$
    hence, $\epsilon = \min\{1\ ,\ 0.8\} = 0.8$
  - The resulting schedule is given by Fig. 9, where B_1 and B_2 are combined to the same block B_1 and B_3 is reindexed.

- $k = 1$
  - Test if B_1 can be improved by applying a left-shift operation to the last idle time

    $f_5(1) - n_2 = 0 - 2 = -2 < 0$

    $\Rightarrow B_1$ is an improvable block.

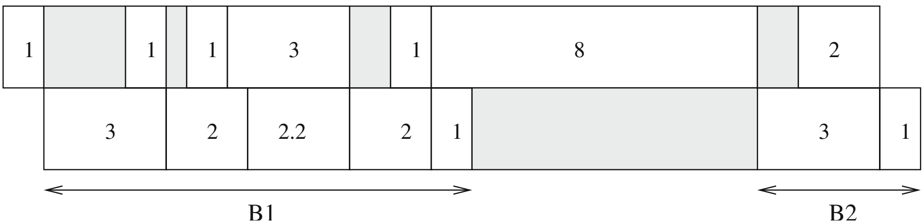

Fig. 9 Placement with $\sum C_i = 83$

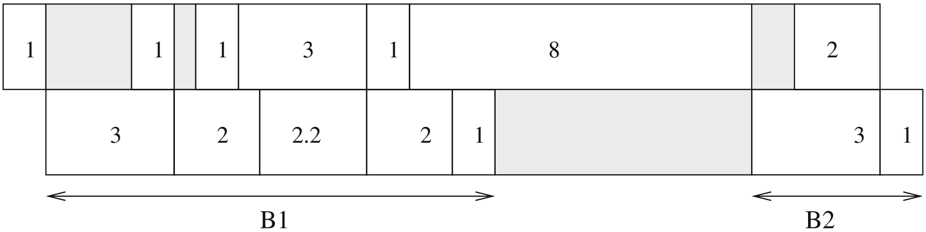

Fig. 10 Placement with $\sum C_i = 81$

– Calculate the value of ϵ.

$$\epsilon_1 = p_{2,4} - p_{1,5} - \sigma_2(4) = 2 - 1 - 0 = 1$$
$$\epsilon_2 = \frac{\gamma_1(1)}{1+f_5(1)} = \frac{8-6}{1+0} = 4$$
$$\text{hence, } \epsilon = \min\{1\ ,\ 4\} = 1$$

– The result schedule is given by Fig. 10 with $\sum_{i=1}^{7} C_i = 81$.

- $k = 1$

– Test if B_1 can be improved by applying a left-shift operation to the last idle time.

$$3\,f_3(1) + 2\,f_4(1)(1 + f_3(1)) + f_5(1)(1 + f_3(1))(1 + f_4(1)) - n_2$$
$$= 0.75 + 0.625 - 2 = -0.2625 < 0$$
$$\Rightarrow B_1 \text{ is an improvable block.}$$

– Calculate the value of ϵ.

$$\epsilon_1 = p_{2,2} - p_{1,3} - \sigma_3(1) = 2 - 1 - 0.8 = 0.2$$
$$\epsilon_2 = \frac{\gamma_1(1)}{1 + f_3(1) + f_4(1)(1 + f_3(1)) + f_5(1)(1 + f_3(1))(1 + f_4(1))}$$
$$= \frac{6}{1.5625} = 3.84$$
$$\text{hence, } \epsilon = \min\{0.2\ ,\ 3.84\} = 0.2$$

– The result schedule is given by Fig. 11 with $\sum_{i=1}^{7} C_i = 80.875$.

- $k = 1$

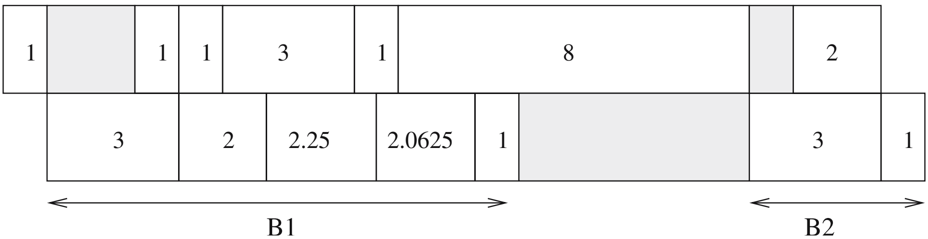

Fig. 11 Placement with $\sum C_i = 80.875$

– Test if B_1 can be improved by applying a left-shift operation to the last idle time.

$$\begin{aligned}&4f_2(1)+3f_3(1)(1+f_2(1))+2f_4(1)(1+f_2(1))(1+f_3(1))+f_5(1)\\&\quad\times(1+f_2(1))(1+f_3(1))(1+f_4(1))-n_2=2+1.125+0.9375-2\\&=2.0625>0\\&\Rightarrow B_1 \text{ is a stable block.}\end{aligned}$$

- $k=2$

  – Test if B_2 can be improved by applying a left-shift operation to the last idle time.

$$\begin{aligned}&f_2(2)-n_3=0.5-0=0.5>0\\&\Rightarrow B_2 \text{ is a stable block.}\end{aligned}$$

All blocks are stable, the optimal placement is given by Fig. 11, with a total completion time equal to 80.875.

6 Conclusion

In this paper, we propose an optimal placement algorithm that minimizes the total completion time criterion for a given order of tasks. In this paper we used general nondecreasing functions for the deterioration of the task processing times on the second machine. As shown, the method compares favorably with linear programming in the special case of linear deterioration functions.

Our algorithm works for two-machine problems. The linear programming method for the linear case may be extended to the m-machine case. However, finding the optimal placement for non linear deterioration functions and $m \geq 3$ machines seems to be difficult and would require other techniques.

References

1. Alidaee, B.: Single machine scheduling with nonlinear cost functions. Comput. Oper. Res. **18**, 317–322 (1991)
2. Chen, Z.-L.: Parallel machine scheduling with time dependent processing times. Discrete Appl. Math. **70**, 81–93 (1996)
3. Finke, G., Jiang, H.: A variant of the permutation flow shop model with variable processing times. Discrete Appl. Math. **76**, 123–140 (1997)
4. Espinouse, M.-L., Finke, G., Jiang, H.: General flowshop models: job dependent capacities, job overlapping and deterioration. Int. Trans. Oper. Res. **9**, 399–414 (2002)
5. Gupta, J.N.D., Gupta, S.K.: Single facility scheduling with nonlinear processing times. Comput. Ind. Eng. **14**, 387–393 (1988)
6. Janiak, A.: General flow-shop scheduling with resource constraints. Int. J. Prod. Res. **26**, 1089–1103 (1988)
7. Janiak, A.: Minimization of resource consumption under a given deadline in two-processor flow-shop scheduling problem. Inf. Process. Lett. **32**, 101–112 (1989)
8. Mosheiov, G.: $\wedge$-shaped policies to schedule deteriorating jobs. J. Oper. Res. Soc. **47**, 1184–1191 (1996)
9. Mosheiov, G.: Multi-machine scheduling with linear deterioration. INFOR **36**, 205–214 (1998)
10. Mosheiov, G.: Complexity analysis of job-shop scheduling with deteriorating jobs. Discrete Appl. Math. **117**, 195–209 (2002)

11. Sriskandarajah, C., Goyal, S.K.: Scheduling of two-machine flowshop with processing time linearly dependent on job waiting-time. J. Oper. Res. Soc. **40**, 907–921 (1989)
12. Sriskandarajah, C., Wagneur, E.: Hierarchical control of the two processor flow-shop with state-dependent processing time: complexity analysis and approximate algorithms. INFOR **29**, 193–205 (1991)
13. Wagneur, E., Sriskandarajah, C.: Optimal control of a class of deds: flow-shops with state-dependent processing times. Discret. Event Dyn. Syst., Theory Appl. **3**, 397–425 (1993)
14. Wagneur, E., Sriskandarajah, C.: The two-machine permutation flow-shops with state-dependent processing times. Nav. Res. Logist. **40**, 697–717 (1993)