Computers & Operations Research 36 (2009) 3031 – 3040

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

# Scheduling of a two-stage differentiation flowshop to minimize weighted sum of machine completion times

T.C.E. Cheng[a], B.M.T. Lin[b,*], Y. Tian[b]

[a]Department of Logistics and Maritimes Studies, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
[b]Institute of Information Management, Department of Information and Finance Management, National Chiao Tung University, Hsinchu 300, Taiwan

ARTICLE INFO

Available online 20 February 2009

*Keywords:*
Flowshop
Machine completion time
Approximation algorithm
Performance ratio

ABSTRACT

This paper considers the problem of scheduling a two-stage flowshop that consists of a common critical machine in stage one and two independent dedicated machines in stage two. All the jobs require processing first on the common critical machine. Each job after completing its critical operation in stage one will proceed to the dedicated machine of its type for further processing in stage two. The objective is to minimize the weighted sum of stage-two machine completion times. We show that the problem is strongly NP-hard, and develop an $O(n^3)$ polynomial time algorithm to solve the special case where the sequences of both types of jobs are given. We also design an approximation algorithm with a tight performance ratio of $\frac{4}{3}$ for the general case.

## 1. Introduction

Flowshop scheduling is one of the most widely studied topics in the scheduling literature [6,23]. This broad topic contains many different settings and special cases, reflecting a wide range of applications. In this paper, we consider a special kind of the two-stage flowshop, where all the products require processing first on a common critical machine in the first stage, after which each product proceeds to a dedicated machine for further processing in the second stage. We call this model the two-stage differentiation flowshop and present the msachine configuration of such a flowshop model in Fig. 1. Many manufacturing environments that produce multiple final products are extensions of this basic model. In particular, this model can be applied to describe the production setting for delayed product differentiation, which is one of the approaches taken to achieve mass customization [4,26]. Kyparisis and Koulamas [15] remarked that "applications of the proposed flowshop model are encountered in manufacturing settings, where all jobs must first go through the same main process, and then they require a finishing operation special to the job". For a specific application, consider a production line for furniture manufacturing (e.g., chairs). The main bodies of the chairs are manufactured in the first stage. Several different head-supports are then assembled in the second stage. Thus, each type of chair products proceeds to a different dedicated machine in the second stage. Clearly, the second stage may be an assembly operation as described above, a painting operation (painting the product in one of several colors), or a packaging operation (wrapping up the product in one of several packages), etc. Another application is pottery production. The main glazing process is performed in the first stage. Several heating processes to produce different figures or surface effects are applied in the second stage. In other words, each type of pottery products proceeds to a corresponding dedicated machine for the required baking process after being glazed. The second stage may consist of re-glazing, various thermal treatments, or packaging.

The machine setting studied in this paper is similar to the hybrid flowshop, where multiple parallel machines are available at each stage and jobs can be processed by any of the machines. The reader is referred to Linn and Zhang [17] for a review of scheduling in hybrid flowshops. In the differentiation model considered in this study, stage two consists of two machines, and jobs are routed to only the dedicated machines of their types. When there is only one type of jobs, the differentiation model reduces to the classical two-machine flowshop proposed by Johnson [13]. The model studied in this paper probably first appeared in Herrmann and Lee [9], in which three objectives, namely the makespan, the number of tardy jobs and the maximum tardiness, were investigated and the corresponding problems were shown to be strongly NP-hard. For the problem to minimize the makespan with two fixed sequences for the two types of jobs, they transformed it into the problem to minimize the maximum lateness such that Jackson's earliest due date (EDD) rule [11] can solve the makespan problem in $O(n \log n)$ time. Given an arbitrary number of stage-two machines in a job shop, Drobouchevitch and Strusevich [5] designed a heuristic algorithm to minimize the makespan with a performance ratio of $\frac{3}{2}$. Kyparisis and Koulamas

* Corresponding author. Tel.: +886 3 5131472; fax: +886 3 5729915.
*E-mail address:* bmtlin@mail.nctu.edu.tw (B.M.T. Lin).

0305-0548/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2009.02.001

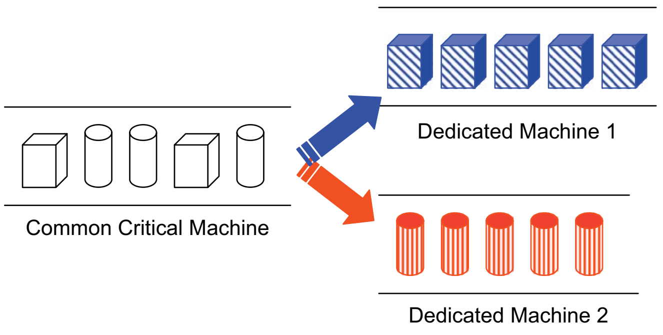


**Fig. 1.** Machine configuration.

[15] investigated the model with $m$ types of jobs that need to be processed on $m$ dedicated machines in the second stage, subject to a technical constraint, called the block assumption, whereby jobs of the same type must be processed contiguously on the critical machine in the first stage. The authors proposed an $O(m(n\log n+\log m))$ algorithm to minimize the makespan. Subsequently, Moshelov and Yovel [20] made several comments on Kyparisis and Koulamas's algorithm and improved the time complexity to $O(n\log n)$ under the assumption that $m\leqslant n$. Cheng and Kovalyov [2] incorporated setup times on the common machine whenever the processing of jobs on it is switched from one type to the other. They proposed a dynamic programming algorithm that is polynomial in the number of jobs for makespan minimization. A model that exhibits a reverse production flow has $m$ dedicated machines for $m$ types of jobs installed in stage one and contains a critical machine that is common for all the jobs in stage two. With regard to makespan minimization, the two symmetric models are equivalent. Specifically, the reverse model with $m=2$ types of jobs was independently studied by Lin [16], Neumytov and Sevastyanov[21] and Oğuz et al. [22].

The goal of makespan minimization, as an internal management metric, is to maximize machine utilization. Another objective is to minimize the weighted total job completion time ($\sum w_iC_i$), which, as a metric of WIP inventory or customer's waiting time, is based upon job completion times. The objective to minimize the weighted sum of machine completion times studied, in this paper, is non-classical. It aims at attaining better machine utilization where different machines may have different operating costs. This objective borrows the concept of the weighted total job completion time, but it generalizes the goal of makespan minimization that focuses on internal management. Bagga [1] first studied the situation where the machine (rental) cost depends on the duration between a machine starts processing its jobs and when it finishes processing all the jobs. Ho and Gupta [10] investigated permutation flowshop scheduling with dominant machines to minimize the total machine completion time. Fondrevelle et al. [7] investigated this objective in a multi-stage flowshop incorporating the constraint of minimum and maximum time lags between successive operations. They showed that the classical two-stage ($F2$) problem can be solved using Johnson's rule and that the problem with three or more stages, i.e., $Fm$, $m\geqslant 3$, is strongly NP-hard. For the parallel-machine model ($Pm$), we can minimize the weighted sum of machine completion times in polynomial time because the problem can be formulated as the classical assignment problem, which is polynomially solvable. The objective of total machine completion time is different from the objective of total machine load, which was introduced by Mosheiov [19] to measure the total time in which the machines are engaged in the production process. The total machine load objective has the potential to deal with the objective of the total completion time as demonstrated in Jeng and Lin [12]. To the best of our knowledge, the three-machine two-stage flowshop scheduling model with the objective of minimizing the weighted sum of machine completion times, denoted by **F(1,2)_WMT**, under study has not been addressed in the scheduling literature. Note that only stage-two machine completion times are involved in the objective function because the completion time of the stage-one machine is fixed once the input instance is given.

The rest of the paper is organized as follows. In Section 2 we present the notation that is used throughout the paper, and give a numerical example to illustrate the problem definition. Section 3 presents the result that the problem under study is strongly NP-hard. We dedicate Section 4 to the development of a polynomial time algorithm for the special case with a fixed sequence for each of the two types of jobs. We develop an approximation heuristic and analyze its performance ratio in Section 5. We present some concluding remarks and suggest some topics for future research in Section 6.

## 2. Problem definition and notation

In this section, we give a formal definition of the **F(1,2)_WMT** problem and define the notation that will be used. We then give an example of two schedules for illustration.

The problem setting of **F(1,2)_WMT** is formally defined as follows. There is a set of $n$ jobs $\mathscr{J}=\{J_1,J_2,\ldots,J_n\}$ available from time zero to be processed on a three-machine two-stage flowshop, where $M_0$ is the stage-one machine and $M_1, M_2$ are two different dedicated machines in the second stage. The jobs belong to two different types: type 1, $\mathscr{J}_1=\{J_1,J_2,\ldots,J_{n_1}\}$ and type 2 $\mathscr{J}_2=\{J_{n_1+1},J_{n_1+2},\ldots,J_{n_1+n_2}\}$ with $n_1+n_2=n$. Each job in $\mathscr{J}_1$ consists of two operations, of which the first is performed on the common critical machine $M_0$, and the second is performed on the dedicated machine $M_1$, as in the classical two-machine flowshop. Similarly, the jobs of $\mathscr{J}_2$ are processed first on the common critical machine $M_0$ and then on the dedicated machine $M_2$. Each machine can process at most one operation at any time, and no preemption is allowed. The goal is to find a schedule that minimizes the weighted sum of machine completion times. Applying the job-interchange argument to jobs of the same type, we can show that it suffices to consider only permutation schedules, i.e., jobs of the same type have the same processing sequence on the critical machine and on their dedicated machine.

**Notation:**

| | |
|---|---|
| $\mathscr{J}=\{J_1,\ldots,J_n\}=\mathscr{J}_1\cup\mathscr{J}_2$: | the set of jobs to be processed; |
| $\mathscr{J}_1=\{J_1,\ldots,J_{n_1}\}$: | the set of type-1 jobs; |
| $\mathscr{J}_2=\{J_{n_1+1},\ldots,J_{n_1+n_2}\}$: | the set of type-2 jobs, where $n_1+n_2=n$; |
| $M_0$: | the stage-one common critical machine; |
| $M_1$: | the stage-two dedicated machine for jobs of $\mathscr{J}_1$; |
| $M_2$: | the stage-two dedicated machine for jobs of $\mathscr{J}_2$; |
| $p_{ki}$: | the processing time on machine $M_k$, $k=0,1,2$, of job $J_i\in\mathscr{J}$; |
| $w_k$: | the weight of machine $M_k$, $k=1,2$; |
| $\sigma=(\sigma_1,\sigma_2,\ldots,\sigma_n)$: | a particular schedule, $\sigma_i\in\mathscr{J}$, $1\leqslant i\leqslant n$; |
| $C_{ki}(\sigma)$: | the completion time of job $J_i\in\mathscr{J}$ on machine $M_k$, $k=0,1,2$, under schedule $\sigma$; |
| $C^{(k)}(\sigma)$: | the completion time of machine $M_k$, $k=0,1,2$, under schedule $\sigma$; |
| $Z(\sigma)=w_1C^{(1)}(\sigma)+w_2C^{(2)}(\sigma)$: | the weighted sum of machine completion times under schedule $\sigma$; |
| $Z^*$: | the optimal weighted sum of machine completion times. |

Four jobs in two types

| Jobs | $J_1$ | $J_2$ | $J_3$ | $J_4$ |
|---|---|---|---|---|
| stage-one | $p_{0,1}=2$ | $p_{0,2}=5$ | $p_{0,3}=4$ | $p_{0,4}=3$ |
| stage-two | $p_{1,1}=4$ | $p_{1,2}=3$ | $p_{2,3}=6$ | $p_{2,4}=2$ |

Assume $w_1 = w_2 = 1$.

Schedule $\sigma_1 = (J_3, J_1, J_2, J_4)$

| Jobs | $J_3$ | $J_1$ | $J_2$ | $J_4$ |
|---|---|---|---|---|
| $C_{0i}$ | 4 | 6 | 11 | 14 |
| $C_{1i}, C_{2i}$ | 10 | 10 | 14 | 16 |

$C^{(1)} = 14, C^{(2)} = 16$ and $Z(\sigma_1) = 14 + 16 = 30.$

Schedule $\sigma_2 = (J_1, J_3, J_4, J_2)$

| Jobs | $J_1$ | $J_3$ | $J_4$ | $J_2$ |
|---|---|---|---|---|
| $C_{0i}$ | 2 | 6 | 9 | 14 |
| $C_{1i}, C_{2i}$ | 6 | 12 | 14 | 17 |

$C^{(1)} = 14, C^{(2)} = 17$ and $Z(\sigma_1) = 14 + 17 = 31.$

**Fig. 2.** Two example schedules with different objective values.

In the above notation, $\sigma$ may be omitted if no ambiguity would arise, and $C_{ki}$ will replace $C_{ki}(\sigma)$. To avoid possible confusion concerning missing operations in the flowshop, we assume that all the processing times $p_{ki}$ are strictly positive. Note that processing times $p_{1i}$ (resp. $p_{2i}$) are only defined for $i \leqslant n_1$ (resp. $i > n_1$). The machine weights $w_k$ are positive, too. Furthermore, if it is not necessary to specify the starting times of the jobs, then a *sequence*, instead of a schedule, of the jobs on machine $M_0$ will be referred to. Fig. 2 shows four jobs, two of each type, to be scheduled. Consider schedules $\sigma_1 = (J_3, J_1, J_2, J_4)$ and $\sigma_2 = (J_1, J_3, J_4, J_2)$. Jobs of each type are sequenced in the Johnson's order in both schedules. The two schedules correspond to different interleaved Johnson's sequences and result in different weighted sums of machine completion times.

## 3. Complexity results

In the following, we show that the $\mathbf{F(1,2)\_WMT}$ problem is NP-hard in the strong sense. We first introduce an optimality property of the studied problem. A job sequence on machine $M_0$ can be divided into blocks, each of which contains jobs of the same type. The following property resolves the sequencing issue within each block.

**Lemma 1.** *There is an optimal schedule in which the jobs of the same block on machine $M_0$ are sequenced by Johnson's rule.*

**Proof.** The validity can be established by applying the job-interchange argument to two consecutive jobs of the same type not following Johnson's order. $\square$

The following NP-hardness proof is based on a reduction from 3-Partition, which is known to be strongly NP-hard [8].

3-Partition: Given a non-negative integer $B$ and a set of $3m$ non-negative integers $A = \{x_1, x_2, \ldots, x_{3m}\}$ with $B/4 < x_i < B/2$ for each $x_i$ and $\sum_{x_i \in A} x_i = mB$, is there a partition $A_1, A_2, \ldots, A_m$ of set $A$ such that for each subset $A_j$, $\sum_{x_i \in A_j} x_i = B$?

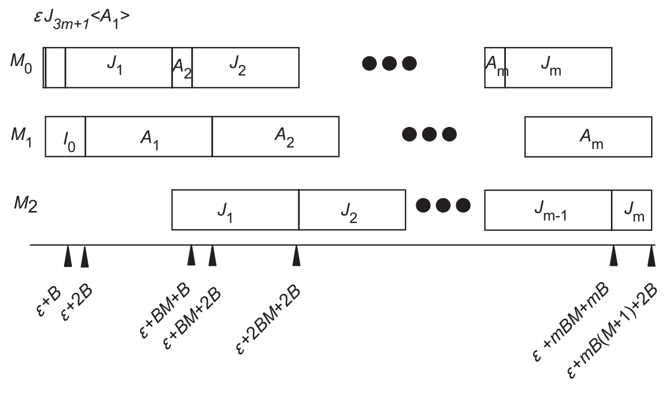


**Fig. 3.** Configuration of the optimal schedule in Theorem 1.

**Theorem 1.** *The* $\mathbf{F(1,2)\_WMT}$ *problem is NP-hard in the strong sense even if the stage-two machines are equally weighted.*

**Proof.** The verification process can be easily done in polynomial time. Therefore, the decision version of the $\mathbf{F(1,2)\_WMT}$ problem is clearly in NP. From a given instance of 3-Partition, we construct the following instance of $\mathbf{F(1,2)\_WMT}$ consisting of $4m + 1$ jobs:

$n_1 = 3m + 1$ type-1 jobs:

$$p_{0i} = x_i, \quad p_{1i} = x_i(M+1), \ i = 1, \ldots, 3m, \ \text{where } M > mB,$$

$$p_{0,3m+1} = \varepsilon, p_{1,3m+1} = 2B \quad \text{where } 0 < \varepsilon < 1.$$

$n_2 = m$ type-2 jobs:

$$p_{0i} = BM, \quad p_{2i} = B(M+1), \ i = 3m+2, \ldots, 4m,$$

$$p_{0,4m+1} = BM, \quad p_{2,4m+1} = 2B.$$

The two dedicated machines have the same weights. Without loss of generality, we assume that $m > 2$. Given a 3-Partition instance and the constructed instance, it can be shown that the answer to 3-Partition is affirmative if and only if there is a schedule for the $\mathbf{F(1,2)_WMT}$problem whose objective value is no greater than $2mB(M+1) + 4B + 2\varepsilon$.

If there is a partition $A_1, A_2, \ldots, A_m$ of set $A$, we construct a schedule by arranging the jobs on machine $M_0$ as

$$(J_{3m+1}, \langle A_1 \rangle, J_{3m+2}, \langle A_2 \rangle, J_{3m+3}, \ldots, J_{4m}, \langle A_m \rangle, J_{4m+1}),$$

where $\langle A_i \rangle$ denotes the type-1 jobs corresponding to the elements of $A_i$. It is easy to verify that the weighted sum of machine completion times is exactly $2mB(M+1) + 4B + 2\varepsilon$. The configuration is shown in Fig. 3.

We now assume that there is a schedule $\sigma$ with $Z(\sigma) \leqslant 2mB(M+1) + 4B + 2\varepsilon$. First, notice that on machine $M_1$ the total processing length of all the jobs is $mB(M+1) + 2B$ and that on machine $M_2$ the total processing length of all the jobs is $(m-1)B(M+1) + 2B$. With these observations, we readily have that the sum of idle times on machine $M_1$ and machine $M_2$ cannot exceed $B(M+1) + 2\varepsilon$. Because all the type-2 jobs have the same operation on machine $M_0$ and job $J_{4m+1}$ has the shortest processing time on $M_2$, we assume that the type-2 jobs are arranged in increasing order of their indices. The validity of this assumption comes from Lemma 1.

Assume that type-2 job $J_{3m+2}$ is scheduled first on $M_0$. Then, an idle time of $BM$ will be incurred on both dedicated machines, resulting in $2BM$ total idle time, a contradiction. Therefore, some type-1 jobs must be scheduled before job $J_{3m+2}$ on $M_0$. Let $\langle A_1 \rangle \cup \{J_{3m+1}\}$ denote the set of type-1 jobs that precede job $J_{3m+2}$ on $M_0$. Before proceeding with the analysis, we note that an idle time of $BM$

is inevitable for the first job $J_{3m+2}$ on $M_2$. By Lemma 1, we assume that type-1 job $J_{3m+1}$ is scheduled first because $p_{0,3m+1} = \varepsilon$. The idle time on $M_1$ is at least $\varepsilon$. Consider the following two cases:

*Case* 1: $\sum_{x_i \in A_1} x_i > B$

The starting time of job $J_{3m+2}$ on machine $M_2$ is $\varepsilon + \sum_{x_i \in A_1} x_i + BM \geqslant \varepsilon + (B+1) + BM > B(M+1) + \varepsilon$. That is, the idle time on $M_2$ is greater than $B(M+1)+\varepsilon$. Combining the idle time $\varepsilon$ on $M_1$, the total idle time on both dedicated machines is greater than $B(M+1)+2\varepsilon$, a contradiction.

*Case* 2: $\sum_{x_i \in A_1} x_i < B$

On machine $M_1$, the completion time of the last job of $\langle A_1 \rangle$ is $\varepsilon + 2B + (M+1)\sum_{x_i \in A_1} x_i$. Because the completion time of job $J_{3m+2}$ on machine $M_0$ is $\varepsilon + \sum_{x_i \in A_1} x_i + MB$, the first type-1 job following $\langle A_1 \rangle$ on machine $M_1$ has an idle time of at least

$$\begin{aligned}
&\left(\varepsilon + \sum_{x_i \in A_1} x_i + MB\right) - \left(\varepsilon + 2B + (M+1)\sum_{x_i \in A_1} x_i\right) \\
&= M\left(B - \sum_{x_i \in A_1} x_i\right) - 2B \\
&\geqslant M - 2B \\
&> mB - 2B \\
&> B.
\end{aligned}$$

Therefore, the total idle time on $M_1$ is greater than $\varepsilon + B$. On the other hand, the total idle time on $M_2$ is at least $\varepsilon + BM$. Hence the sum of the idle times on the two stage-two machines is greater than $B(M+1)+2\varepsilon$, a contradiction.

From the analysis of the above two cases, we know that the equality $\sum_{x_i \in A_1} x_i = B$ must hold. Let the elements corresponding to the type-1 jobs included in $\langle A_1 \rangle$ form a subset $A_1$. When job $J_{3m+2}$ is finished, the completion times of the three machines are $\varepsilon+BM+B$, $\varepsilon+BM+3B$ and $\varepsilon+2BM+2B$, respectively. We can continue the same analysis to obtain sets $A_2, A_3, \ldots, A_m$ and complete the proof. $\square$

Before closing this section, we note that the instance constructed in the proof exhibits three side conditions: (1) The dedicated machines are equally weighted. (2) Agreeable condition: for any jobs $J_i$ and $J_j$ of the same type, if $p_{0i} < p_{0j}$, then $p_{1i} \leqslant p_{1j}$ (type 1) or $p_{2i} \leqslant p_{2j}$ (type 2). (3) Equal-processing-time condition: all the type-2 jobs have the same processing time on $M_0$. In other words, the **F(1,2)_WMT** problem remains computationally intractable even if the input instance satisfies these three assumptions commonly adopted to deal with difficult scheduling problems. The following sections are dedicated to one special case that can be solved in polynomial time, and to the development of an approximation algorithm for the general case.

## 4. Fixed sequences

In this section, we consider a simplified situation where the sequences of both types of jobs are fixed, i.e., two independent predetermined sequences are given. Under this assumption, the problem reduces to finding an interleaved sequence on machine $M_0$ from the two given fixed sequences. The problem is motivated as follows. We have two sets of products to manufacture and each set has its processing sequence predetermined by some job characteristics or technological constraints. The products will be assigned a common resource, i.e., machine time on the stage-one machine, so we need to construct an interleaved sequence of all the products on the common machine. In most scheduling problems, schedules are implicitly implied from sequences. For some problems, it is nevertheless hard to determine an optimal schedule from fixed sequences. In two-machine flowshop scheduling with batch processing, it is not trivial to determine an optimal batching policy of a given sequence to minimize the total completion time. To minimize the makespan in the $F(1,2)$ model, Hermann and Lee [9] reduced the problem to the problem to minimize the maximum lateness, which is solvable in $O(n \log n)$ time. Shafransky and Strusevich [24] studied open shop scheduling to minimize the makespan, subject to a given job sequence on one machine. They investigated several cases and showed them to be NP-hard or polynomially solvable.

We denote the $F(1,2)$ problem with two fixed sequences by $\mathbf{F(1,2)\_WMT}_{\mathit{fixed\_seq}}$. Consider the two schedules in Fig. 2 as an example. Both schedules are obtained from interleaving sequences $(J_1,J_2)$ and $(J_3,J_4)$; however, the weighted sums of machine completion times of the two interleaved sequences $(J_3,J_1,J_2,J_4)$ and $(J_1,J_3,J_4,J_2)$ are different. For notational simplicity, let $(J_1,J_2,\ldots,J_{n_1})$ and $(J_{n_1+1},J_{n_1+2},\ldots,J_n)$ denote the given sequences. Given the two sequences, there are $(n_1+n_2)!/(n_1)!(n_2)!$ possible interleaved sequences. The problem is to develop a solution algorithm to determine an interleaved sequence from such a solution space with an exponential size. Preliminary investigation suggests that the problem is not trivial to solve. At least, no simple method or dispatching rule has been found to deliver optimal solutions.

In this section we explore several structural properties that can help identify promising candidates. The main algorithm will construct at most $n_1(n_2+1)$ schedules, among which an optimal schedule is identified. In particular, for each job $J_i$ of type 1, and for each $l = 0,\ldots,n_2$, the algorithm finds a feasible schedule (if such exists) to minimize the completion time of machine $M_2$. The schedule must satisfy the following two requirements: there are exactly $l$ jobs of type 2 preceding job $J_i$ on $M_0$ and job $J_i$ is the first job of type 1 after which machine $M_1$ has no idle time.

Let $\sigma^*$ be an optimal sequence. We examine the processing of type-1 jobs on machine $M_1$. In the following discussion, when index $i=0$ refers to a job, a dummy job with zero processing time is assumed; when index $i=0$ refers to job completion times, $C_{0,0}=0$ and $C_{1,0}=0$ are assumed. First, we find the largest index $i$ such that $C_{0i} > C_{1,i-1}$ and $C_{0r} \leqslant C_{1,r-1}$ for any $r, i < r \leqslant n_1$. In other words, on machine $M_1$, a non-zero idle time exists before job $J_i$ and jobs $J_i,J_{i+1},\ldots,J_{n_1}$ are processed consecutively without idle time inserted. We further assume that exactly $j$ type-2 jobs $J_{n_1+1},\ldots,J_{n_1+j}$ precede job $J_i$ in $\sigma^*$. Therefore, job $J_i$ has $i+j-1$ predecessors. The above arrangement implies that the completion time of machine $M_1$ is

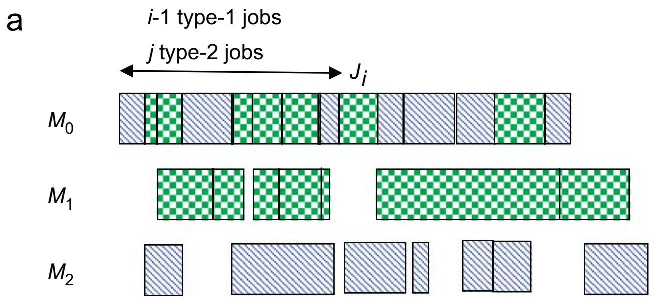
$$C^{(1)}(\sigma^*) = \sum_{r=1}^{i} p_{0r} + \sum_{r=n_1+1}^{n_1+j} p_{0r} + \sum_{r=i}^{n_1} p_{1r}.$$

Given the above arrangement with the fixed completion time $C^{(1)}(\sigma^*)$ of machine $M_1$, we then seek to minimize the completion time $C^{(2)}(\sigma^*)$ of machine $M_2$. The above discussion leads to the following notion.

**Definition.** The ordered pair $(i,j), 1 \leqslant i \leqslant n_1, 0 \leqslant j \leqslant n_2$ is *admissible* if there exists an interleaved sequence $\sigma$ that satisfies the following conditions:

(a) Job $J_i$ is preceded by type-1 jobs $J_1,\ldots,J_{i-1}$ and type-2 jobs $J_{n_1+1},\ldots,J_{n_1+j}$.
(b) $C_{1,i-1}(\sigma) < C_{0,i}(\sigma) = \sum_{r=1}^{i} p_{0r} + \sum_{r=n_1+1}^{n_1+j} p_{0r}$.
(c) For $i < r \leqslant n_1$, $C_{0,r}(\sigma) \leqslant C_{1,r-1}(\sigma)$.

Condition (b) specifies that there is a non-zero idle time on machine $M_1$ before $J_i$. Condition (c) dictates that job $J_i$ is the last job of set $\mathscr{J}_1$ to have idle time before its second-stage operation, i.e., the processing of $J_i,\ldots,J_{n_1}$ on machine $M_1$ is consecutive without idle time inserted. An illustration of the configuration with two schedules is shown in Fig. 4.



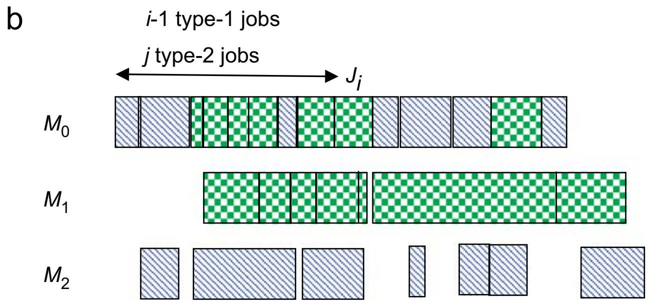


**Fig. 4.** Tow schedules for an admissible ordered pair (i, j).

**Definition.** For the ordered pair $(i,j), 1 \leqslant i \leqslant n_1, 0 \leqslant j \leqslant n_2$, define the *basic* sequence $\sigma(i,j)$ as $(J_1,\dots,J_{i-1},J_{n_1+1},\dots,J_{n_1+j},J_i,\dots,J_{n_1},J_{n_1+j+1},\dots,J_n)$.

**Lemma 2.** *If the basic sequence $\sigma(i,j)$ violates condition* (b) *or condition* (c), *then the ordered pair $(i,j)$ cannot be admissible.*

**Proof.** The starting time $\sum_{r=1}^{i} p_{0r} + \sum_{r=n_1+1}^{n_1+j} p_{0r}$ of job $J_i$ is fixed for the pair $(i,j)$. The subsequence $(J_1,\dots,J_{i-1},J_{n_1+1},\dots,J_{n_1+j})$ clearly provides the smallest completion time of $J_{i-1}$ on $M_1$ for all the possible interleaved sequences derived from $(J_1,\dots,J_{i-1})$ and $(J_{n_1+1},\dots,J_{n_1+j})$. Therefore, if the subsequence $(J_1,\dots,J_{i-1},J_{n_1+1},\dots,J_{n_1+j})$ violates condition (b), then so will all the other interleaved sequences. If condition (c) fails, then on $M_1$ a non-zero idle time exists before some type-1 job $J_{i'}$, $i' > i$, in the basic sequence. The idle time before job $J_{i'}$ can be eliminated only if the idle time before $J_i$ vanishes, which violates condition (b). Therefore, if condition (c) fails, the ordered pair $(i,j)$ cannot be admissible. The proof is complete.

The theme of our solution algorithm now becomes one of determining all the admissible ordered pairs $(i,j)$ using Lemma 2. For each admissible ordered pair, we find an interleaved sequence that minimizes the completion time of machine $M_2$. Finally, we take the minimum of the weighted sum of machine completion times over all the admissible pairs. The minimization of weighted sum of machine-two completion times of the basic sequence consists of two subproblems:

Problem $X$: Minimize the completion time of $J_{n_1+j}$ on machine $M_2$ with condition (b) satisfied, and;

Problem $Y$: Subject to the solution for Problem $X$, minimize the completion time of job $J_n$ on machine $M_2$ with condition (c) satisfied.

For Problem $X$, it suffices to consider the partial sequence $\bar{\sigma}(i,j) = (J_1,\dots,J_{i-1},J_{n_1+1},\dots,J_{n_1+j})$. For $1 \leqslant r \leqslant i-1$, define $\bar{\mathscr{I}}_r$ as the subsequence of type-2 jobs of the subset $\{J_{n_1+1},\dots,J_{n_1+j}\}$ that are scheduled between jobs $J_r$ and $J_{r+1}$. In addition, define $\bar{\mathscr{I}}_0$ as the subsequence of type-2 jobs of $\{J_{n_1+1},\dots,J_{n_1+j}\}$ that are scheduled before $J_1$. The set of jobs in sequence $\bar{\mathscr{I}}_r, 0 \leqslant r \leqslant i-1$, is denoted by $\{\bar{\mathscr{I}}_r\}$. Initially, in subsequence $\bar{\sigma}(i,j)$, we have $\bar{\mathscr{I}}_0 = \bar{\mathscr{I}}_1 = \cdots = \bar{\mathscr{I}}_{i-2} =$ `null`, and $\bar{\mathscr{I}}_{i-1} = (J_{n_1+1},\dots,J_{n_1+j})$. The following procedure transforms subsequence $\bar{\sigma}(i,j)$ so that the completion time of job $J_{n_1+j}$ is minimized, subject to condition (b).

Let $\bar{\sigma}'(i,j)$ be a sequence derived by interleaving the two types of jobs from the partial sequence $\bar{\sigma}(i,j) = (J_1,\dots,J_{i-1},J_{n_1+1},\dots,J_{n_1+j})$ for Problem $X$. Sequence $\bar{\sigma}'(i,j)$ is called feasible if condition (b) is satisfied. Given two feasible sequences $\bar{\sigma}'(i,j)$ and $\bar{\sigma}''(i,j)$, we say that $\bar{\sigma}'(i,j)$ *dominates* $\bar{\sigma}''(i,j)$ if $\bigcup_{l=0}^{r}\{\bar{\mathscr{I}}'_l\} \supseteq \bigcup_{l=0}^{r}\{\bar{\mathscr{I}}''_l\}$ for all $0 \leqslant r \leqslant i-1$. The schedule of Fig. 4(a) dominates that of Fig. 4(b).

**Observation 1.** *Giventwo feasible sequences $\bar{\sigma}'(i,j)$ and $\bar{\sigma}''(i,j)$ for Problem X, if $\bar{\sigma}'(i,j)$ dominates $\bar{\sigma}''(i,j)$, then the completion times of $J_{n_1+j}$ on machine $M_0$ and machine $M_2$ with respect to $\bar{\sigma}'(i,j)$ are no greater than those with respect to $\bar{\sigma}''(i,j)$, i.e., $C_{0,n_1+j}(\bar{\sigma}'(i,j)) \leqslant C_{0,n_1+j}(\bar{\sigma}''(i,j))$ and $C_{2,n_1+j}(\bar{\sigma}'(i,j)) \leqslant C_{2,n_1+j}(\bar{\sigma}''(i,j))$.*

The observation reveals that Problem $X$ reduces to finding a feasible sequence that dominates all the feasible ones. The following procedure is designed to find such a sequence. In the algorithm, the symbols "⊕" and "⊖" denote sequence concatenation and sequence deletion, respectively.

```
PROCEDURE X($\bar{\sigma}(i,j)$)
Step 1: Set $\bar{\mathscr{I}}_0 = \bar{\mathscr{I}}_1 = \cdots = \bar{\mathscr{I}}_{i-2} =$ null, and $\bar{\mathscr{I}}_{i-1} = (J_{n_1+1},\dots,J_{n_1+j})$.
Step 2: Set $r = 0; s = n_1 + 1$.
Step 3: **While** $(r < i-1)$ **and** $(s \leqslant n_1 + j)$ **do**
            **if** moving $J_s \in \bar{\mathscr{I}}_{i-1}$ into $\bar{\mathscr{I}}_r$ will not violate condition (b)
              **then** $\bar{\mathscr{I}}_r := \bar{\mathscr{I}}_r \oplus (J_s)$; $\bar{\mathscr{I}}_{i-1} := \bar{\mathscr{I}}_{i-1} \ominus (J_s)$; $s := s+1$;
              **else** $r := r+1$.
Step 4: **Return** sequence $\bar{\sigma}(i,j) = (\bar{\mathscr{I}}_0, J_1, \bar{\mathscr{I}}_1, \dots, \bar{\mathscr{I}}_{i-2}, J_{i-1}, \bar{\mathscr{I}}_{i-1})$.
```

**Lemma 3.** PROCEDURE X *for Problem X produces a feasible sequence that dominates all the feasible sequences.*

**Proof.** First, note that during the course of execution of PROCEDURE X, the produced sequence $\bar{\sigma}(i,j)$ satisfies condition (b), so feasibility is maintained.

Let $\bar{\sigma}'(i,j) = (\bar{\mathscr{I}}'_0, J_1, \bar{\mathscr{I}}'_1, \dots, \bar{\mathscr{I}}'_{i-2}, J_{i-1}, \bar{\mathscr{I}}'_{i-1})$ be a feasible sequence not dominated by sequence $\bar{\sigma}(i,j)$. There must be a smallest $r$ such that $\bigcup_{l=0}^{r}\{\bar{\mathscr{I}}_l\} \subset \bigcup_{l=0}^{r}\{\bar{\mathscr{I}}'_l\}$. Let $J_x$ be the type-2 job of $\bigcup_{l=0}^{r}\{\bar{\mathscr{I}}'_l\} \backslash \bigcup_{l=0}^{r}\{\bar{\mathscr{I}}_l\}$ with the smallest index. By the logic of the **if** test in Step 3, augmenting $\bar{\mathscr{I}}_r$ with $J_x$ will cause infeasibility, and thus $\bar{\sigma}'(i,j)$ cannot be feasible. Therefore, there cannot be such a feasible sequence $\bar{\sigma}'(i,j)$. □

With regard to the computing time, it is easy to see that condition (b) is examined for at most $O(n_1 + n_2) = O(n)$ times. A naïve approach can be deployed to examine condition (b) for the current sequence under consideration by a simple $O(n)$ loop, thus, resulting in an overall $O(n^2)$ time. The concept of composite jobs, which was proposed by Kurisu [14], can be used to reduce to constant time the time required by checking condition (b) for a single sequence. Applications of composite jobs can be found in, e.g., Monma [18], Sidney [25], and Cheng and Lin [3].

Assume that the string of jobs $(J_1,\dots,J_{i-1})$ is to be processed on machines $M_0$ and $M_1$ as in the two-machine flowshop without any other jobs inserted in the string. By the concept of composite job, the processing of the job string $(J_1,\dots,J_{i-1})$ can be replaced by a single composite job, denoted by $J_{[1:i-1]}$ in the sense that the string and its corresponding composite job have the same total idle time on $M_1$. To define composite job $J_{[1:i-1]}$, we first consider jobs $J_{i-2}$ and $J_{i-1}$. Composite job $J_{[i-2:i-1]}$ is defined by letting

$$p_{0,[i-2:i-1]} = p_{0,i-2} + \max\{0, p_{0,i-1} - p_{1,i-2}\},$$

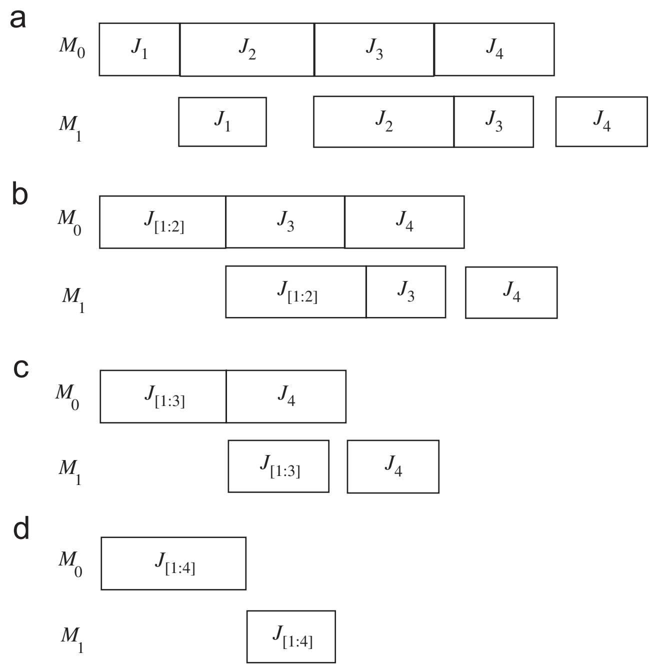


**Fig. 5.** Formation of composite jobs.

and

$$p_{1,[i-2:i-1]} = \max\{0, p_{1,i-2} - p_{0,i-1}\} + p_{1,i-1}.$$

Combining $J_{i-3}$ and $J_{[i-2:i-1]}$, we can then define $J_{[i-3:i-1]}$ by letting

$$p_{0,[i-3:i-1]} = p_{0,i-3} + \max\{0, p_{0,[i-2:i-1]} - p_{1,i-3}\},$$

and

$$p_{1,[i-3:i-1]} = \max\{0, p_{1,i-3} - p_{0,[i-2:i-1]}\} + p_{1,[i-2:i-1]}.$$

Therefore, composite jobs $J_{[i-2:i-1]}, J_{[i-3:i-1]}, \ldots, J_{[1:i-1]}$ can be successively found in a backward manner in $O(n)$ time. Composite jobs $J_{[r:s]}, 1 \leqslant r < s \leqslant i-1$, can be similarly derived in $O(n^2)$ time. This preprocessing step will be elaborated in the analysis of another algorithm to be discussed later.

**Lemma 4** (Kurisu [14]). *Sequences* $(J_1, \ldots, J_{i-1})$ *and* $(J_1, \ldots, J_{r-1}, J_{[r:s]}, J_{s+1}, \ldots, J_{i-1})$, $1 \leqslant r < s \leqslant i-1$, *have the same total idle time on machine* $M_1$.

The schedules shown in Fig. 5 demonstrate how composite jobs are formed and the total idle time on the stage-two machine remains unchanged. Assume that all the composite jobs $J_{[r:i-1]}, 1 \leqslant r \leqslant i-1$ are derived when the input instance is given. The required time for the preprocessing step is $O(n^2)$, which will not be included in the computing time of PROCEDURE X.

**Lemma 5.** *Given the composite jobs defined*, PROCEDURE X *solves Problem X in* $O(n)$ *time.*

**Proof.** Lemma 3 confirms that PROCEDURE X produces a feasible schedule that minimizes the completion times of job $J_{n_1+j}$ on machines $M_0$ and $M_2$. To complete the proof, it suffices to show that checking condition (b) for each iteration of Step 3 requires only $O(1)$ time.

Let $T_0$ and $T_1$ be the completion times of machines $M_0$ and $M_1$ immediately after consideration of appending a particular type-2 job $J_s$–$\bar{\mathscr{I}}_r$. If not terminated, depending on whether or not the insertion of $J_s$ into $\bar{\mathscr{I}}_r$ is feasible, the algorithm will proceed to examining either $J_{s+1}$ and $\bar{\mathscr{I}}_r$ or $J_s$ and $\bar{\mathscr{I}}_{r+1}$. For the former case, job $J_{s+1}$ and composite job $J_{[r+1:i-1]}$ are scheduled from $T_0$ and $T_1$. On machine $M_1$, the idle time immediately preceding $J_{[r+1:i-1]}$ is $\eta = \max\{0, T_0 + p_{0,s+1} + p_{0,[r+1:i-1]} - T_1\}$. Therefore, the completion time of job $J_{i-1}$ is

$$C_{1,i-1} = T_1 + \eta + \sum_{l=r+1}^{i-1} p_{1l}.$$

If, on the other hand, $J_s$ and $\bar{\mathscr{I}}_{r+1}$ are considered in the next iteration, then $\eta = \max\{0, T_0 + p_{0,s} + p_{0,[r+2:i-1]} - T_1\}$ and

$$C_{1,i-1} = T_1 + \eta + \sum_{l=r+2}^{i-1} p_{1l}.$$

Having the value of $C_{1,i-1}$, we can readily examine condition (b) in constant time. The proof is complete. □

Based upon the sequence $\bar{\sigma}(i,j)$ produced by PROCEDURE X, we can then proceed to the second part to deal with Problem *Y*. We start with the sequence

$$\tilde{\sigma}(i,j) = \bar{\sigma}(i,j) \oplus (J_i, \ldots, J_{n_1}, J_{n_1+j+1}, \ldots, J_n).$$

For $i \leqslant r \leqslant n_1 - 1$, define $\tilde{\mathscr{I}}_r$ as the subsequence of type-2 jobs of $\{J_{n_1+j+1}, \ldots, J_n\}$ that are scheduled between jobs $J_r$ and $J_{r+1}$. The sequence of type-2 jobs arranged after $J_{n_1}$ is denoted by $\tilde{\mathscr{I}}_{n_1}$. Initially, in the subsequence $\tilde{\sigma}(i,j)$, we have $\tilde{\mathscr{I}}_i = \tilde{\mathscr{I}}_{i+1} = \cdots = \tilde{\mathscr{I}}_{n_1-1} = \texttt{null}$, and $\tilde{\mathscr{I}}_{n_1} = (J_{n_1+j+1}, \ldots, J_n)$. As the development and analysis are similar to those for Problem *X*, we omit the details of the proofs.

PROCEDURE Y

```
Step 1: Set $\tilde{\mathscr{I}}_i = \tilde{\mathscr{I}}_{i+1} = \cdots = \tilde{\mathscr{I}}_{n_1-1} = \texttt{null}$, and $\tilde{\mathscr{I}}_{n_1} = (J_{n_1+j+1}, \ldots, J_n)$.
Step 2: Set $\tilde{\sigma}(i,j) = \bar{\sigma}(i,j) \oplus (J_i, \tilde{\mathscr{I}}_i, \ldots, \tilde{\mathscr{I}}_{n_1-1}, J_{n_1}, \tilde{\mathscr{I}}_{n_1})$.
Step 3: Set $r = i; s = n_1 + j + 1$.
Step 4: While $(r < n_1)$ and $(s \leqslant n)$ do
            if moving $J_s \in \tilde{\mathscr{I}}_{n_1}$ into $\tilde{\mathscr{I}}_r$ does not violate condition (c)
              then $\tilde{\mathscr{I}}_r := \tilde{\mathscr{I}}_r \oplus (J_s); \tilde{\mathscr{I}}_{n_1} := \tilde{\mathscr{I}}_{n_1} \ominus (J_s); s := s + 1$;
              else $r := r + 1$.
Step 5: Return sequence $\tilde{\sigma}(i,j) = \bar{\sigma}(i,j) \oplus (J_i, \tilde{\mathscr{I}}_i, \ldots, \tilde{\mathscr{I}}_{n_1-1}, J_{n_1}, \tilde{\mathscr{I}}_{n_1})$.
```

**Lemma 6.** PROCEDURE Y *for Problem Y produces a feasible sequence that dominates all the feasible sequences.*

**Lemma 7.** *Given the defined composite jobs*, PROCEDURE Y *solves Problem Y in* $O(n)$ *time.*

The result concerning the determination of the earliest completion time of machine $M_2$ for an admissible pair $(i,j)$ is summarized in the following.

**Lemma 8.** *Given two fixed sequences and an admissible pair* $(i,j)$, *a schedule yielding the smallest completion time of machine* $M_2$ *can be found in* $O(n)$ *time.*

**Proof.** If the pair $(i,j)$ is admissible, then the basic schedule $(i,j)$ is obtained by sequence concatenation. PROCEDURE X is invoked on the prefix partial sequence $\bar{\sigma}(i,j)$, which is initially given by $(J_1, \ldots, J_{i-1}, J_{n_1+1}, \ldots, J_{n_1+j})$. Then we invoke PROCEDURE Y on sequence $\tilde{\sigma}(i,j) = \bar{\sigma}(i,j) \oplus (J_i, \ldots, J_{n_1}, J_{n_1+j+1}, \ldots, J_n)$. After execution of the two procedures, we obtain a sequence in which the type-2 jobs are scheduled as early as possible such that the completion time of $M_2$ is minimum while maintaining conditions (b) and (c). The execution of PROCEDURES X and Y takes $O(n)$ time. This completes the proof. □

Consider the following instance with five jobs of each type.

$$n_1 = n_2 = 5$$

| Jobs | $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ | $J_6$ | $J_7$ | $J_8$ | $J_9$ | $J_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $p_{i0}$ | 3 | 5 | 9 | 8 | 2 | 2 | 4 | 11 | 3 | 2 |
| $p_{i1}, p_{i2}$ | 5 | 6 | 8 | 7 | 3 | 6 | 5 | 13 | 2 | 1 |

Subsequences $(J_1,J_2,J_3,J_4,J_5)$ and $(J_6,J_7,J_8,J_9,J_{10})$ are given. Admissible ordered pairs are $(3,1),(3,2),(3,3),(3,4),(3,5),(4,1),(4,2),(4,3),(4,4),(4,5),(5,1),(5,2),(5,3),(5,4),(5,5)$.

All the other ordered pairs are inadmissible. Taking ordered pair $(3,3)$ as an example, we have the basic sequence $(J_1,J_2,J_6,J_7,J_8,J_3,J_4,J_5,J_9,J_{10})$. Through PROCEDURE X, jobs $J_6,J_7,J_8$ are moved with condition (b) preserved. The sequence output by PROCEDURE X is $(J_6,J_7,J_8,J_1,J_2,J_3,J_4,J_5,J_9,J_{10})$. Through PROCEDURE Y, jobs $J_9,J_{10}$ are moved with condition (c) preserved. The final output is $(J_6,J_7,J_8,J_1,J_2,J_3,J_4,J_9,J_{10},J_5)$. The Gantt charts of the above sequences are shown in Fig. 6.

The following algorithm integrates the above ingredients to determine the interleaved sequence that minimizes the weighted sum of machine completion times.

ALGORITHM FIXED-SEQUENCES

```
Input:  Two sequences $(J_1,J_2,\ldots,J_{n_1})$ and $(J_{n_1+1},J_{n_1+2},\ldots,J_n)$.
Output: Interleaved sequence $\sigma^*$ that minimizes the weighted sum
        of machine completion times.
Step 1: Set $Z^* = \infty$;
Step 2: Derive $\sum_{r=1}^{i} p_{0i}$ and $\sum_{r=i}^{n_1} p_{0i}$ for $1 \leqslant i \leqslant n_1$, and $\sum_{r=n_1+1}^{i} p_{0i}$
        for $1 \leqslant i \leqslant n_2$.
Step 3: For $i = 1$ to $n_1$ define $J_{[i:i]} = J_i$.
Step 4: For $s = n_1$ down to 2 do
          for $r = s-1$ down to 1 do
            Derive composite job $J_{[r:s]}$ from $J_r$ and $J_{[r+1:s]}$.
Step 5: For each pair $(i,j), 1 \leqslant i \leqslant n_1, 0 \leqslant j \leqslant n_2$, do
          if basic sequence $\sigma(i,j)$ is admissible, then
            Call PROCEDURE X and PROCEDURE Y;
            if $Z(\tilde{\sigma}(i,j)) < Z^*$, then $\sigma^* = \tilde{\sigma}(i,j); Z^* = Z(\sigma^*)$.
Step 6: Return $\sigma^*$ and $Z^*$.
```

**Theorem 2.** ALGORITHM FIXED-SEQUENCES *solves problem* $\mathbf{F(1,2)\_WMT}_{fixed\_seq}$ *in* $O(n^3)$ *time.*

**Proof.** The algorithm inspects all the possible ordered pairs, for each of which PROCEDURES X and Y produce a sequence with the earliest completion time of $M_2$. Taking the minimum weighted sum of machine completion times among all the feasible ordered pairs, ALGORITHM FIXED-SEQUENCES returns the optimal solution. As for the computing time, the preprocessing part, Steps 1-4, of ALGORITHM FIXED-SEQUENCES takes $O(n^2)$ time. There are $O(n^2)$ ordered pairs $(i,j)$ to examine in Step 5 and examining a pair requires $O(n)$ time. Therefore, the overall running time is $O(n^3)$. □

In the following we extend the result given in Theorem 2. Recall the agreeable condition defined in Section 3. We consider the reverse of the agreeable condition:

**rev_agr**: For any jobs $J_i, J_j \in \mathscr{J}_1$, if $p_{0i} < p_{0j}$, then $p_{1i} \geqslant p_{1j}$, and for any jobs $J_i, J_j \in \mathscr{J}_2$, if $p_{0i} < p_{0j}$, then $p_{2i} \geqslant p_{2j}$.

The case satisfying condition **rev_agr** is denoted by $\mathbf{F(1,2)\_WMT}_{rev\_agr}$. To solve this case, we re-index the type-1 jobs such that if $i < j$, then "$p_{0i} < p_{0j}$" or "$p_{0i} = p_{0j}$ and $p_{1i} \geqslant p_{1j}$". Break ties arbitrarily. Type-2 jobs are similarly re-indexed.

**Lemma 9.** *For problem* $\mathbf{F(1,2)\_WMT}_{rev\_agr}$, *there is an optimal schedule in which the jobs of each type are sequenced in increasing order of their indices.*

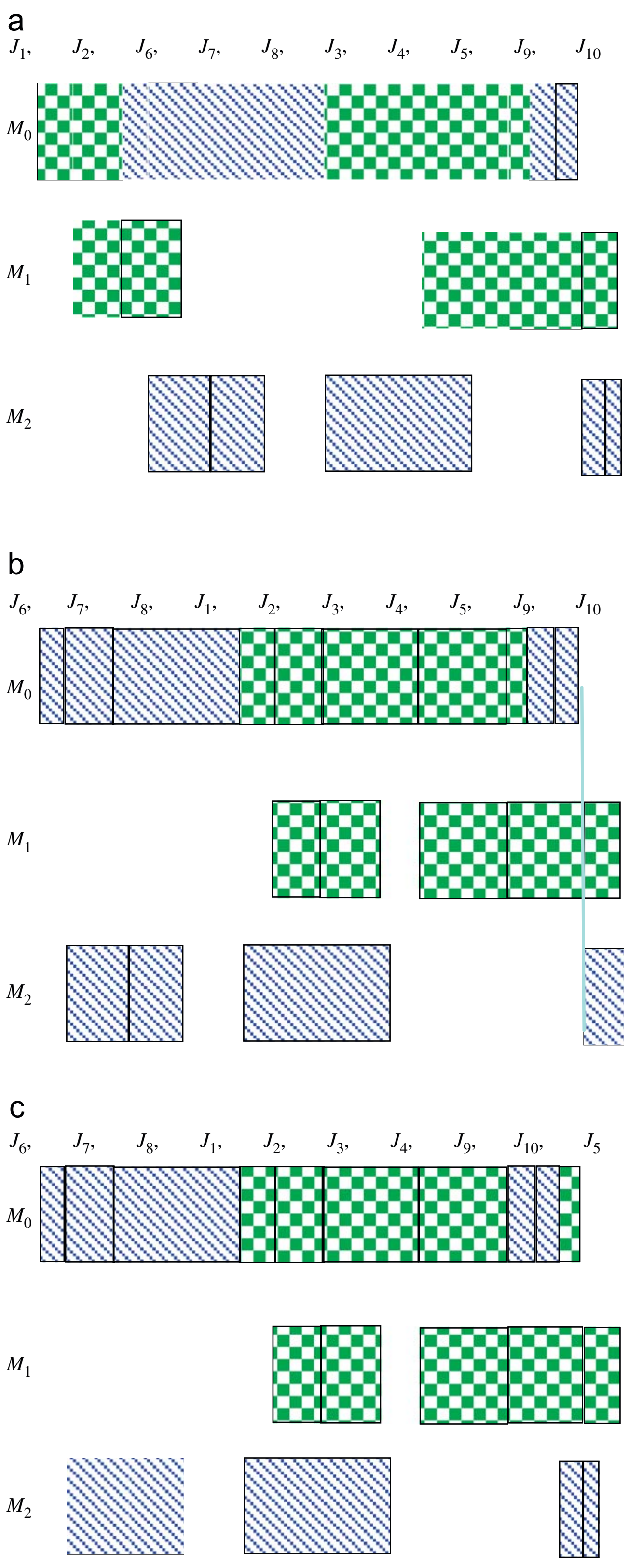


**Fig. 6.** Execution of PROCEDURES X and Y on ordered pair (3,3).

**Proof.** Assume that in some optimal solution for $\mathbf{F(1,2)\_WMT}_{rev\_agr}$, there are jobs not sequenced as specified. Let $J_i, J_j, 1 \leqslant i < j \leqslant n_1$, be the first two type-1 jobs such that job $J_j$ precedes job $J_i$. We exchange the positions of the two jobs. The standard job-interchange argument is applied in two phases, based on operations rather than on jobs. First, we interchange their stage-one operations and leave their stage-two operations unchanged. That is, $p_{0i}$ and $p_{1j}$ constitute one job, and $p_{0j}$ and $p_{1i}$ constitute another job. It is clear that the completion of any job of $\mathscr{J}\backslash\{J_i,J_j\}$ will not be deferred. Next we interchange the positions of the dedicated operations of the two jobs. Similarly, no completion time of any other jobs will increase. Therefore, the schedule derived from swapping the positions of jobs $J_i$ and $J_j$ will not increase the makespan. Continuing the swapping process, if necessary, we can obtain a schedule in which the type-1 jobs are sequenced in increasing order of their indices. The same line of reasoning can be applied to type-2 jobs. This completes the proof. $\square$

By Theorem 2, the following result immediately follows.

**Corollary 1.** *Problem* $\mathbf{F(1,2)\_WMT}_{rev\_agr}$ *can be solved in* $O(n^3)$ *time.*

## 5. Approximation algorithm

This section is devoted to the development of a heuristic and analysis of its worst-case performance. Given a minimization problem and an approximation algorithm $H$, the *performance ratio* of the algorithm on a given instance is defined as the ratio of the solution value $Z_H$ given by the algorithm to the optimal value $Z^*$, i.e., $Z_H/Z^*$. If for any instance, an algorithm attains a performance ratio less than or equal to $r$, then it is called an $r$-approximate algorithm for the studied problem. We give a $\frac{4}{3}$-approximate algorithm for the problem $\mathbf{F(1,2)\_WMT}$ in this section.

First, we derive a lower bound on the objective value for analysis. Jobs of $\mathscr{J}_1$ and jobs of $\mathscr{J}_2$ are independently sequenced by Johnson's rule. A lower bound is then obtained from the resulting weighted sum of completion times:

$$LB = w_1 C_{JS}^{(1)} + w_2 C_{JS}^{(2)},$$

where $C_{JS}^{(1)}$ and $C_{JS}^{(2)}$ are the makespan (completion times of the two dedicated machines) of the two Johnson's sequences. An extra term can be added to the bound if we consider the interaction between the jobs of two different types. Let $P_1 = \sum_{i=1}^{n_1} p_{0i}$ and $P_2 = \sum_{i=n_1+1}^{n} p_{0i}$. Assume that the last job on machine $M_0$ belongs to type 1. If we consider only type-1 jobs, then for a particular sequence $\sigma$, the difference between the completion times of machines $M_1$ and $M_0$ is no less than $C_{JS}^{(1)} - P_1$. The insertion of the processing of type-2 jobs on $M_0$ will delay the completion of the last type-1 job in $\sigma$ by $P_2$. Consequently, the makespan of sequence $\sigma$ will increase by at least $\max\{0, P_2 - (C_{JS}^{(1)} - P_1)\}$. Adding this term to the previous lower bound, we get

$$\begin{aligned} LB_1 &= w_1 C_{JS}^{(1)} + w_2 C_{JS}^{(2)} + w_1 \max\{0, P_1 + P_2 - C_{JS}^{(1)}\} \\ &= w_2 C_{JS}^{(2)} + w_1 \max\{C_{JS}^{(1)}, P_1 + P_2\}. \end{aligned}$$

On the other hand, if the last job on machine $M_0$ belongs to type 2, then we have the following lower bound:

$$LB_2 = w_1 C_{JS}^{(1)} + w_2 \max\{C_{JS}^{(2)}, P_1 + P_2\}.$$

The heuristic algorithm presented below is mainly based upon application of Johnson's algorithm to the two job sets $\mathscr{J}_1$ and $\mathscr{J}_2$, which we will show to be a $\frac{4}{3}$-approximate algorithm for the problem $\mathbf{F(1,2)\_WMT}$.

Heuristic H:

*Step* 1: Let $\sigma(\mathscr{J}_1)$ (resp. $\sigma(\mathscr{J}_2)$) be the Johnson's sequence for $\mathscr{J}_1$ (resp. $\mathscr{J}_2$).

*Step* 2: **If** $w_2P_1 \leqslant w_1P_2$, **then return** $\sigma(\mathscr{J}_1) \oplus \sigma(\mathscr{J}_2)$; otherwise, **return** $\sigma(\mathscr{J}_2) \oplus \sigma(\mathscr{J}_1)$.

The computing time required by Heuristic H is dominated by the sorting stage for deriving the two Johnson's sequences. Therefore, the running time is $O(n \log n)$. Denote by $Z_H$ the weighted sum of machine completion times of the schedule produced by Heuristic H. In the following, we analyze the performance ratio of Heuristic H. Without loss of generality, we assume that $w_2P_1 \leqslant w_1P_2$. The case of $w_2P_1 \geqslant w_1P_2$ is symmetric. By the algorithm, all the type-1 jobs sequenced by Johnson's rule are scheduled first as a block. The machine completion times of the schedule produced by Heuristic H are $C_{JS}^{(1)}$ and $P_1 + C_{JS}^{(2)}$, respectively. Therefore, the weighted sum is given by $Z_H = w_1 C_{JS}^{(1)} + w_2(P_1 + C_{JS}^{(2)})$.

**Theorem 3.** Heuristic H *is a* $\frac{4}{3}$*-approximate algorithm for* $\mathbf{F(1,2)\_WMT}$ *and the bound is tight.*

**Proof.** To establish the correctness of the result, we focus on estimating $\max\{Z_H/LB_1, Z_H/LB_2\}$ because

$$\min\{LB_1, LB_2\} \leqslant Z^*.$$

Let $P_1 = \alpha P_2$ for some positive $\alpha$.

*Case* 1: $LB_1 \leqslant LB_2$

In this case, $Z_H/LB_1$ is considered. The assumption that $w_2P_1 \leqslant w_1P_2$ implies the inequality $\alpha w_2 \leqslant w_1$. Because $w_2P_1 = w_2\alpha P_2 \leqslant \alpha w_2 C_{JS}^{(2)}$, we also have $C_{JS}^{(2)} \geqslant P_1/\alpha$. Therefore,

$$\begin{aligned} \frac{Z_H}{LB_1} &= \frac{w_1 C_{JS}^{(1)} + w_2P_1 + w_2 C_{JS}^{(2)}}{w_2 C_{JS}^{(2)} + w_1 \max\{C_{JS}^{(1)}, P_1 + P_2\}} \\ &\leqslant 1 + \frac{w_2P_1}{w_2 C_{JS}^{(2)} + w_1 \max\{C_{JS}^{(1)}, P_1 + P_2\}} \\ &\qquad (\text{because } C_{JS}^{(1)} \leqslant \max\{C_{JS}^{(1)}, P_1 + P_2\}) \\ &\leqslant 1 + \frac{w_2P_1}{w_2 C_{JS}^{(2)} + w_1P_1 + w_1P_2} \\ &\qquad (\text{because } \max\{C_{JS}^{(1)}, P_1 + P_2\} \geqslant P_1 + P_2) \\ &\leqslant 1 + \frac{w_2P_1}{w_2P_1/\alpha + \alpha w_2P_1 + w_2P_1} \\ &\qquad (\text{because } C_{JS}^{(2)} \geqslant P_1/\alpha,\ w_1 \geqslant \alpha w_2,\ w_1P_2 \geqslant w_2P_1) \\ &= 1 + \frac{\alpha}{\alpha^2 + \alpha + 1}. \end{aligned}$$

*Case* 2: $LB_1 \geqslant LB_2$

Considering $Z_H/LB_2$, we have

$$\begin{aligned} \frac{Z_H}{LB_2} &= \frac{w_1 C_{JS}^{(1)} + w_2P_1 + w_2 C_{JS}^{(2)}}{w_1 C_{JS}^{(1)} + w_2 \max\{C_{JS}^{(2)}, P_1 + P_2\}} \\ &\leqslant 1 + \frac{w_2P_1}{w_1 C_{JS}^{(1)} + w_2 \max\{C_{JS}^{(2)}, P_1 + P_2\}} \\ &\qquad (\text{because } C_{JS}^{(2)} \leqslant \max\{C_{JS}^{(2)}, P_1 + P_2\}) \\ &\leqslant 1 + \frac{w_2P_1}{w_1 C_{JS}^{(1)} + w_2P_1 + w_2P_2} \\ &\qquad (\text{because } \max\{C_{JS}^{(2)}, P_1 + P_2\} \geqslant P_1 + P_2). \end{aligned}$$

We relate $w_1 C_{JS}^{(1)}$ and $w_2P_2$ in the denominator to $w_2P_1$ in the numerator. Inequalities $w_1 \geqslant \alpha w_2$ and $C_{JS}^{(1)} \geqslant P_1$ imply $w_1 C_{JS}^{(1)} \geqslant \alpha w_2 P_1$.

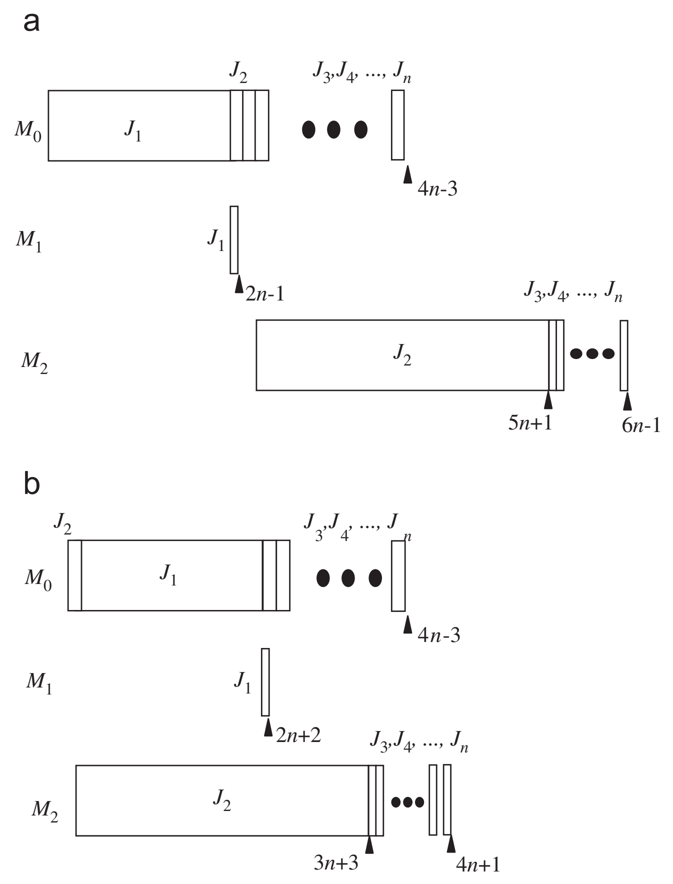


**Fig. 7.** (a): Schedule produced by HEURISTIC H. (b) Optimal schedule.

Therefore,

$$\frac{Z_H}{LB_2} \leqslant 1 + \frac{w_2P_1}{w_1C^{(1)}_{JS} + w_2P_1 + w_2P_2}$$
$$\leqslant 1 + \frac{w_2P_1}{\alpha w_2P_1 + w_2P_1 + w_2P_1/\alpha}$$
$$= 1 + \frac{\alpha}{\alpha^2 + \alpha + 1}.$$

From the discussion of Cases 1 and 2, a performance ratio of $\alpha/(\alpha^2 + \alpha + 1)$ is obtained. We further note that the function $\alpha/(\alpha^2 + \alpha + 1)$ attains its maximum of $\frac{1}{3}$ when $\alpha = 1$. Thus, $Z_H/Z^* \leqslant Z_H/\min\{LB_1, LB_2\} \leqslant \frac{4}{3}$.

Consider the following instance of $n$ jobs to establish the asymptotic tightness of this bound. In the instance $n_1 = 1$ and $n_2 = n - 1$. Assume that the stage-two machines are equally weighted. The single type-1 job is defined as $p_{01} = 2n - 2,\ p_{11} = 1$. The type-2 jobs are defined as $p_{02} = 3,\ p_{22} = 3n$, and $p_{0i} = 2,\ p_{2i} = 1$ for $3 \leqslant i \leqslant n$. Because $P_1 = 2n - 2 < P_2 = 2(n - 2) + 3$, HEURISTIC H produces the sequence $(J_1, J_2, J_3, \ldots, J_n)$ (Fig. 7(a)) with $Z_H = (2n - 1) + (6n - 1) = 8n - 2$. The optimal sequence for the instance is $(J_2, J_1, J_3, \ldots, J_n)$ (Fig. 7(b)) with $Z^* = (2n + 2) + (4n + 1) = 6n + 3$. As $n$ approaches infinity, the ratio $Z_H/Z^*$ approaches $\frac{4}{3}$, implying that $\frac{4}{3}$ is, indeed, a tight bound. □

Considering the results of Section 4, we know that a better heuristic can be designed by applying ALGORITHM FIXED-SEQUENCES to the two independent Johnson's sequences for the two types of jobs. However, there exist no clear structural properties for the analysis of the performance ratio.

## 6. Conclusion

In this paper, we investigated a scheduling problem in a two-stage flowshop that has a common critical machine in stage one and two independent dedicated machines in stage two. The objective is to minimize the weighted sum of machine completion times. We showed that the problem is strongly NP-hard even under three common assumptions on the input instances. Given two fixed sequences for the two types of jobs, we developed an $O(n^3)$ algorithm to determine an optimal interleaved sequence. We also designed a heuristic for the problem and analyzed its performance ratio as $\frac{4}{3}$.

For further research, it will be of interest to generalize our results to a constant number $(> 2)$ of stage-two machines. The case with a variable number of stage-two machines is also interesting. A third possible research issue is analysis of the performance ratio addressed at the end of Section 5. Under some circumstances, the machines could have different processing modes, e.g., batch machines can be considered. Incorporating batch machines, with continuous batches or simultaneous batches, into the differentiation model will give rise to several interesting research problems.

## Acknowledgments

We are grateful to the reviewers for their helpful comments on an earlier version of the paper. Lin and Tian were supported in part by the National Science Council of Taiwan under Grant number 97-2923-H-009-001-MY3.

## References

[1] Bagga PC. Sequencing in a rental solution. Journal of the Canadian Operational Research Society 1969;7:152–3.
[2] Cheng TCE, Kovalyov MY. An exact algorithm for batching and scheduling two part types in a mixed shop: a technical note. International Journal of Production Economics 1998;55(10):53–6.
[3] Cheng TCE, Lin BMT. Johnson's rule, composite jobs and the relocation problem. European Journal of Operational Research 2009;192(3):1008–13.
[4] Da Silveira G, Borenstein D, Fogliatto FS. Mass customization: literature review and research directions. International Journal of Production Economics 2001;72(1):1–13.
[5] Drobouchevitch IG, Strusevich VA. Heuristics for the two-stage job shop scheduling problem with a bottleneck machine. European Journal of Operational Research 2000;123:229–40.
[6] Dudek RA, Panwalkar SS, Smith ML. The lessons of flowshop scheduling research. Operations Research 1992;40:7–13.
[7] Fondrevelle J, Oulamara A, Portmann M-C. Permutation flowshop scheduling problems with time lags to minimize the weighted sum of machine completion times. International Journal of Production Economics 2008;112(1):168–76.
[8] Garey MR, Johnson DS. Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco, CA: Freeman; 1979.
[9] Herrmann JW, Lee C-Y. Three-machine look-ahead scheduling problems. Research Report no. 92–93, Department of Industrial Engineering, University of Florida, FL; 1992.
[10] Ho JC, Gupta JND. Flowshop scheduling with dominant machines. Computers & Operations Research 1995;22(2):237–46.
[11] Jackson JR. Scheduling a production line to minimize maximum lateness. Research Report 43, Management science research report, University of California, Los Angeles; 1955.
[12] Jeng AAK, Lin BMT. A note on parallel-machine scheduling with deteriorating jobs. Journal of the Operational Research Society 2007;58:1099–102.
[13] Johnson SM. Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1954;1:61–7.
[14] Kurisu T. Two-machine scheduling under required precedence among jobs. Journal of the Operations Research Society of Japan 1976;19:1–13.
[15] Kyparisis GJ, Koulamas C. Flow shop and open shop scheduling with a critical machine and two operations per job. European Journal of Operational Research 2000;127(1):120–5.
[16] Lin BMT. The strong NP-hardness of two-stage flowshop scheduling problem with a common second-stage machine. Computers & Operations Research 1999;27(6):695–8.
[17] Linn R, Zhang W. Hybrid flow shop scheduling: a survey. Computers & Industrial Engineering 1999;37(1–2):57–61.
[18] Monma CL. The two-machine maximum flow time problem with series-parallel precedence constraints: an algorithm and extensions. Operations Research 1979;27(4):792–8.
[19] Mosheiov G. Multi-machine scheduling with linear deterioration. INFOR 1998;36(4):205–14.

[20] Mosheiov G, Yovel U. Comments on Flow shop and open shop scheduling with a critical machine and two operations per job. European Journal of Operational Research 2004;157(1):257–61.
[21] Neumytov YD, Sevastyanov SV. Approximation algorithm with tight bound for three-machine counter-routes problem. Upravlyaemye Sistemy 1993;31:53–65 (in Russian).
[22] Oğuz C, Lin BMT, Cheng TCE. Two-stage flowshop scheduling problem with a common second-stage machine. Computers & Operations Research 1997;24(12):1169–74.
[23] Reisman A, Kumar A, Motwani J. Flowshop scheduling/sequencing research, 1952–1994: a statistical review of the literature. IEEE Transactions on Engineering Management 1997;44:316–29.
[24] Shafransky YM, Strusevich VA. The open shop scheduling problem with a given sequence of jobs on one machine. Naval Research Logistics 1998;45:705–31.
[25] Sidney JB. The two-machine maximum flow time problem with series parallel precedence relations. Operations Research 1979;27(4):782–91.
[26] Simchi-Levi D, Kaminsky P, Simchi-Levi E. Designing and Managing the Supply Chain: Concepts, Strategies and Case Studies. MA: McGraw-Hill; 2000.