Two-stage assembly-type flowshop batch scheduling problem subject to a fixed job sequence

FJ Hwang and BMT Lin*

Institute of Information Management, Department of Information and Finance Management, National Chiao Tung University, Hsinchu, Taiwan

This paper discusses a two-stage assembly-type flowshop scheduling problem with batching considerations subject to a fixed job sequence. The two-stage assembly flowshop consists of m stage-1 parallel dedicated machines and a stage-2 assembly machine which processes the jobs in batches. Four regular performance metrics, namely, the total completion time, maximum lateness, total tardiness, and number of tardy jobs, are considered. The goal is to obtain an optimal batching decision for the predetermined job sequence at stage 2. This study presents a two-phase algorithm, which is developed by coupling a problem-transformation procedure with a dynamic program. The running time of the proposed algorithm is $O(mn+n^5)$, where n is the number of jobs.

Journal of the Operational Research Society (2012) **63,** 839–845. doi:10.1057/jors.2011.90
Published online 21 September 2011

Keywords: assembly flowshop; batch scheduling; fixed sequence; dynamic programming

1. Introduction

In machine or shop scheduling, the issues of fixed-job-sequence scheduling are worthy of consideration for some problems where schedules cannot be immediately determined by given sequences of jobs. For these problems, establishing an optimal schedule from a predetermined sequence could be non-trivial because another decision such as batching, interleaving, and idle time insertion would be needed for optimality (Hwang, 2011). This study addresses a two-stage assembly-type flowshop scheduling problem with batching considerations subject to a fixed job sequence. The considered two-stage assembly flowshop is a generalisation of the classical two-machine flowshop studied by Johnson (1954). A typical example of the assembly-type flowshop scheduling concerns the production of fire engines (Lee *et al*, 1993). A fire engine comprises three major components, that is, the body, the chassis, and the engine. These components are respectively produced by three parallel dedicated machines at stage 1 and then delivered to a stage-2 assembly line for final assembly operations.

The problem under study is stated as follows. From time zero onwards, a set of n jobs is to be processed in a two-stage flowshop equipped with $m+1$ machines. At stage 1, there are m parallel dedicated machines which independently produce the m components for jobs. Then these components are transferred to the stage-2 assembly line for assembly operation. Each job consists of $m+1$ specific operations to be executed respectively on the m stage-1 parallel dedicated machines and the stage-2 assembly machine. At stage 2, the assembly operations are processed in batches. The sequential-batch (or sum-batch) model with batch availability is adopted. The setup on the assembly machine is non-anticipatory and assumed to be a constant setup time. Batch availability indicates that jobs of the same batch complete at the same time, when processing of the latest job in this batch has been finished. In the sum-batch model, the processing time of a batch is defined as the sum of the setup time and the processing times of all jobs belonging to this batch. The non-anticipatory setup implies that a setup can start only after all the components of the jobs in the same batch arrive at stage 2 and the stage-2 assembly machine is not occupied. We also assumed that the centralised decision-making policy is adopted. Namely, all the m parallel dedicated machines comply with the sequencing and batching decisions determined by the assembly site. In this study, we address four regular objective functions, that is, the total completion time, the maximum lateness, the total tardiness, and the number of tardy jobs. Following Lin *et al* (2007) who consider the criterion of makespan for the same machine setting, the problem under study is denoted by $(m+1)\mathrm{MAF}|m\delta\rightarrow\beta$, *s-batch*, *fixed_seq*$|\gamma$,

*Correspondence: BMT Lin, Institute of Information Management, Department of Information and Finance Management, National Chiao Tung University, 1001 University Road, Hsinchu, Taiwan.
E-mail: bmtlin@mail.nctu.edu.tw

where $(m+1)$MAF stands for the $(m+1)$-machine assembly flowshop, $m\delta \rightarrow \beta$ for a two-stage system with m stage-1 discrete processors and a stage-2 batch processor, *s-batch* for sum-batch model, *fixed_seq* for fixed job sequence, and $\gamma \in \{\sum C_j, L_{\max}, \sum T_j, \sum U_j\}$.

Regarding complexity status, problem $(m+1)$MAF$|m\delta \rightarrow \beta$, *s-batch*$|\gamma$ is hard to tackle due to the base fact that $F2\|\gamma$ is known to be strongly NP-hard for $\gamma \in \{\sum C_j, L_{\max}, \sum T_j, \sum U_j\}$. Owing to the strong NP-hardness, this paper investigates the setting with a fixed job sequence, $(m+1)$MAF$|m\delta \rightarrow \beta$, *s-batch*, *fixed_seq*$|\gamma$. The assumption of a fixed job sequence can be justified from both practical and theoretical aspects (Hwang, 2011). Practical considerations include technological or managerial decisions (Shafransky and Strusevich, 1998) and First-Come-First-Served principle (Hwang *et al*, 2010). The main theoretical motivation comes from the development of local search and meta-heuristic algorithms for the NP-hard scheduling problem where a schedule cannot be readily induced from a job sequence. In the local search procedure, an efficient approach to the problem subject to a fixed job sequence can be exploited to evaluate the candidate job sequences. Other justifications of the fixed-job-sequence assumption can also be found in previous studies (Herrmann and Lee, 1992; Cheng and Wang, 1999; Cheng *et al*, 2000b; Lin and Cheng, 2006; Lin *et al*, 2007; Ng and Kovalyov, 2007; Potts and Whitehead, 2007; Cheng *et al*, 2009; Hwang and Lin, 2011; Lin and Cheng, 2011; Lin and Hwang, 2011).

The remainder of this paper is organised as follows. A brief literature review of flowshop scheduling with batching considerations subject to a fixed job sequence is given in Section 2. Formal problem definition is provided in Section 3. In Section 4, we present a two-phase algorithm for the studied problem. The last section summarises our results and provides suggestions for future research.

2. Review on batch scheduling in flowshop with a fixed-job-sequence

Flowshop scheduling usually becomes NP-hard when batching is taken into account (please refer to Potts and Van Wassenhove (1992), Allahverdi *et al* (1999), Potts and Kovalyov (2000), Cheng *et al* (2000a) for comprehensive surveys of scheduling models with batching or setup times). In the following, we review batch scheduling in flowshops subject to fixed job sequences. Potts and Kovalyov (2000) indicated that dynamic programming is a viable approach to tackling single-machine batching problem where the sequencing and batching decisions can be decoupled. However, for shop models the fixed job sequence is necessary for the design of dynamic programs. Considering a given job sequence in a two-machine flowshop comprising a stage-1 discrete processor and a stage-2 batch processor, Lin and Cheng (2005) developed an $O(n^2)$ algorithm to minimise the makespan. For the fixed-job-sequence problem in a two-machine flowshop with both batch processors, Cheng *et al* (2000b) presented an $O(n^3)$ algorithm for makespan minimisation. An $O(n^5)$ algorithm was proposed by Hwang *et al* (2010) with the same machine setting for the minimisation of total completion time. For makespan minimisation, an $O(n^{5m-7})$ dynamic programming algorithm for the generalised m-machine environment was presented by Ng and Kovalyov (2007). With regard to the fabrication and assembly scheduling in a two-machine flowshop, each job consists of three components: a common component and a unique component which are both executed on machine 1, and an assembly component which is executed on machine 2 after the above two components are completed. Common components of all jobs are executed in batches, each of which is preceded by the same setup time. For makespan minimisation in the identical common component case, Cheng and Wang (1999) proposed an $O(n^4)$ algorithm to optimally batch the jobs sequenced according to Johnson's rule. For the constant assembly time case, another $O(n^3)$ algorithm was developed for optimally batching the jobs sequenced according to the agreeable processing time condition. Actually, the $O(n^4)$ algorithm proposed by Cheng and Wang (1999) can be utilised for the general case. As for the assembly-type flowshop batching problem, Lin *et al* (2007) proposed an $O(n^2)$ dynamic programming algorithm for problem 3MAF$|2\delta \rightarrow \beta$, *s-batch*, *fixed_seq*$|C_{\max}$. Their algorithm can be generalised to problem $(m+1)$MAF$|m\delta \rightarrow \beta$, *s-batch*, *fixed_seq*$|C_{\max}$, and has an $O(mn^2)$ running time. The complexity results of related fixed-job-sequence flowshop batching problems are summarised in Table 1. Most of these cited results centre around the minimisation of makespan. The four objective functions considered in this study exhibit more complicated structures due to the min-sum objectives and/or the due date constraints.

3. Problem definition and preliminaries

In this section, a formal problem definition is provided for the base case with $m=2$. Generalisation to the case with arbitrary m parallel dedicated machines will be given in the next section.

Assume without loss of generality that a given sequence of jobs $(1, 2, \ldots, n)$ is to be processed in a two-stage assembly flowshop with two dedicated parallel machines, M_a and M_b, at stage 1 and one assembly machine M_2 at stage 2. Each job j consists of three operations to be processed on machine M_a, M_b and M_2, respectively. The corresponding processing times are respectively $p_{a,j}$, $p_{b,j}$ and $p_{2,j}$. Each job j is also characterised by a due date d_j.

Table 1 Complexity results of related fixed-job-sequence flowshop batching problems

Problem	*Complexity*	*Reference*
$F2\|\delta\rightarrow\beta$, *s-batch*, *fixed_seq*$\|C_{\max}$	O(n^2)	Lin and Cheng (2005)
$F2\|$*s-batch*, *fixed_seq*$\|C_{\max}$	O(n^3)	Cheng *et al* (2000b)
$Fm\|$*s-batch*, *fixed_seq*$\|C_{\max}$	O(n^{5m-7})	Ng and Kovalyov (2007)
$F2\|$*s-batch*, *fixed_seq*$\|\sum C_j$	O(n^5)	Hwang *et al* (2010)
$F2\|$*s-batch*, (c, u_j, a_j), *fixed_seq*$\|C_{\max}$	O(n^4)	Cheng and Wang (1999)
$F2\|$*s-batch*, (c_j, u_j, a), *fixed_seq*$\|C_{\max}$	O(n^3)	Cheng and Wang (1999)
$F2\|$*s-batch*, (c_j, u_j, a_j), *fixed_seq*$\|C_{\max}$	O(n^4)	*
3MAF$\|2\delta\rightarrow\beta$, *s-batch*, *fixed_seq*$\|C_{\max}$	O(n^2)	Lin *et al* (2007)
$(m+1)$MAF$\|m\delta\rightarrow\beta$, *s-batch*, *fixed_seq*$\|C_{\max}$	O(mn^2)	†

*The result is derived from Cheng and Wang (1999); †The result is derived from Lin *et al* (2007).

To facilitate our discussion, we define

$$p_{a,[i:j]} = \sum_{l=i}^{j} p_{a,l},$$

$$p_{b,[i:j]} = \sum_{l=i}^{j} p_{b,l},$$

$$p_{2,[i:j]} = \sum_{l=i}^{j} p_{2,l}, \text{ and}$$

$$d_{[i:j]} = \min_{l\in\{i,i+1,\ldots,j\}} d_l.$$

After both stage-1 operations of job j are completed, these two produced component parts of job j are transferred to stage 2 for assembly. Machine M_2 processes the jobs in batches with a non-anticipatory constant setup time s. The processing length of a batch on machine M_2 is defined as the setup time s plus the processing times of all jobs included in this batch. The kth batch formed at stage 2 is denoted by B_k. The objective is to optimally batch the given job sequence for the minimisation of the addressed performance measure. Consider the following instance with $n=6$ for illustration: $(p_{a,1}, p_{b,1}, p_{2,1})=(2,1,1)$, $(p_{a,2}, p_{b,2}, p_{2,2})=(1,3,3)$, $(p_{a,3}, p_{b,3}, p_{2,3})=(4,2,2)$, $(p_{a,4}, p_{b,4}, p_{2,4})=(1,3,1)$, $(p_{a,5}, p_{b,5}, p_{2,5})=(2,2,2)$, $(p_{a,6}, p_{b,6}, p_{2,6})=(5,3,3)$, $(d_1, d_2, d_3, d_4, d_5, d_6)=(8,10,14,19,20,22)$, and $s=1$. Assume that the batching decision is to group jobs {1, 2} into batch B_1, jobs {3, 4, 5} into batch B_2, and {6} into batch B_3. As illustrated in Figure 1, the obtained schedule has $\sum C_j=90$, $L_{\max}=3$, $\sum T_j=4$ and $\sum U_j=2$.

Notation

n	number of jobs
m	number of stage-1 parallel dedicated machines

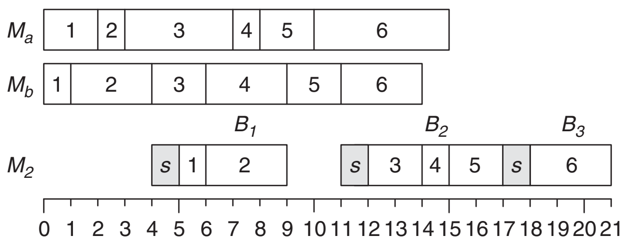

Figure 1 Example schedule.

$p_{a,j}$	processing time of job j on the stage-1 machine M_a
$p_{b,j}$	processing time of job j on the stage-1 machine M_b
$p_{2,j}$	processing time of job j on the stage-2 machine M_2
d_j	due date of job j
$p_{a,[i:j]}$	sum of processing times $p_{a,i}, p_{a,i+1}, \ldots, p_{a,j}$, that is, $p_{a,[i:j]}=\sum_{l=i}^{j} p_{a,l}$
$p_{b,[i:j]}$	sum of processing times $p_{b,i}, p_{b,i+1}, \ldots, p_{b,j}$, that is, $p_{b,[i:j]}=\sum_{l=i}^{j} p_{b,l}$
$p_{2,[i:j]}$	sum of processing times $p_{2,i}, p_{2,i+1}, \ldots, p_{2,j}$, that is, $p_{2,[i:j]}=\sum_{l=i}^{j} p_{2,l}$
$d_{[i:j]}$	minimum of due dates $d_i, d_{i+i}, \ldots, d_j$, that is, $d_{[i:j]}=\min_{l\in\{i,i+1,\ldots,j\}} d_l$
s	batch setup time
B_k	the kth batch formed at stage 2
$C_{1,j}$	completion time of job j on dummy machine M_1, that is, $C_{1,j}=\max\{p_{a,[1:j]}, p_{b,[1:j]}\}$
$T_j(t)$	tardiness of job j which completes at time t, that is, $T_j(t)=\max\{0, t-d_j\}$
$U_j(t)$	tardiness status of job j which completes at time t, that is, $U_j(t)=1$ for $t > d_j$, and $U_j(t)=0$ for $t\leqslant d_j$

4. Two-phase algorithm

In this section, we present a two-phase algorithm to solve problems $(m+1)$MAF$|m\delta\rightarrow\beta$, *s-batch*, *fixed_seq*$|\gamma$ for $\gamma \in\{\sum C_j, L_{\max}, \sum T_j, \sum U_j\}$. The first phase is a pre-processing procedure for transforming the studied problem to a two-machine flowshop batching problem subject to a fixed job sequence. Notice that given the instance input (numerical values and a job sequence), the schedule of jobs at stage 1 can be settled. Then problem $(m+1)$ MAF$|m\delta\rightarrow\beta$, *s-batch*, *fixed_seq*$|\gamma$ can be transformed to problem $F2|\delta\rightarrow\beta$, *s-batch*, *fixed_seq*$|\gamma$ by mapping the stage-1 schedule of the original problem to the stage-1 discrete processor in the transformed problem. Consider the base case $m=2$. As illustrated in Figure 2, the schedule of a *dummy* discrete processor M_1 can be mapped from that of the two parallel dedicated machines M_a and M_b by setting the completion time of job j on M_1 as $C_{1,j}=\max\{p_{a,[1:j]}, p_{b,[1:j]}\}$. Then the studied problem is transformed to a two-machine flowshop problem with a discrete processor M_1 at stage 1 and a batch machine M_2 at stage 2. If m is constant, then O(n) time is needed for

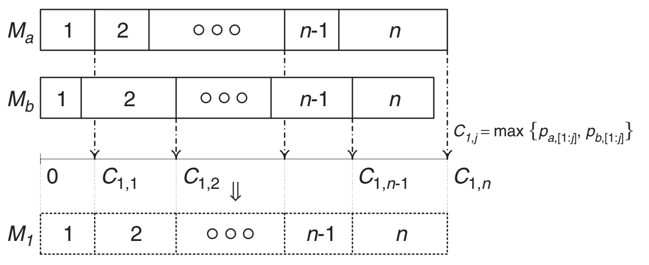

Figure 2 Mapping of the schedules of M_a and M_b to that of M_1.

this transformation procedure. For the general case of arbitrary m, the first phase requires O(mn) time.

In the second phase, problem $F2|\delta\rightarrow\beta$, *s-batch*, *fixed_seq*$|\gamma$ is solved by a dynamic program. The difficulty in the design of a polynomial time dynamic program arises from the potential conflicts between the makespan and the considered objective functions. A subschedule of the first j jobs that minimises the considered performance measure may have a comparatively large makespan, which will worsen the performance measure of the remaining n–j jobs. To resolve the problem, a dynamic program incorporating one more state variable to specify possible makespans can be developed. Nevertheless, the time complexity of the dynamic program designed by this approach will be pseudo-polynomial time. The technique devised in this study is to introduce a fixed number of jobs or positional indices to specify makespans.

For a partial schedule, a maximal (by inclusion) sequence of stage-2 batches processed consecutively without inserted idle times on the stage-2 machine is called a *block*. The last block of a partial schedule is called *critical block*. A state (i, i_1, j, k) specifies all the partial schedules of the first j jobs $1, 2, \ldots, j$ which satisfy the following three conditions:

1. i and j are respectively the first and the last jobs of the critical block,
2. k is the number of batches within the critical block, and
3. i_1 is the last job of the first batch within the critical block.

The structure of a partial schedule in state (i, i_1, j, k) is shown in Figure 3. Note that each state can be associated with several partial schedules having different objective function values. Denote by $g(i, i_1, j, k)$ the minimum objective function value attained from among those partial schedules associated with state (i, i_1, j, k) for $1\leqslant i\leqslant i_1\leqslant j\leqslant n$ and $\lceil(j-i_1)/j\rceil+1\leqslant k\leqslant j-i_1+1$. A partial schedule in state (i, i_1, j, k) with value $g(i, i_1, j, k)$ dominates all other partial schedules in the same state in the sense that it contributes the minimum value to the objective function value among those of all partial schedules in this state. The development of the proposed dynamic program is based

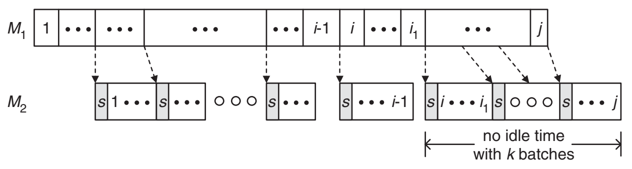

Figure 3 Illustration of a partial schedule in state (i, i_1, j, k).

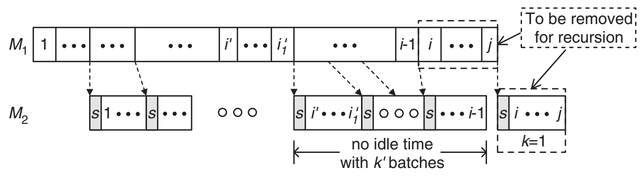

Figure 4 Forward recursion of the case $k=1$

upon forward recursions by batches, that is, the last batch of the partial schedule is removed for each recursion. The dynamic programming formulation is given in a pseudo-code-like fashion. To facilitate notation, denote by $T_j(t)=\max\{0, t-d_j\}$ the tardiness of job j which completes at time t in a particular schedule. Also, its tardiness status is indicated by the binary variable

$$U_j(t) = \begin{cases} 1, & \text{if } t>d_j; \\ 0, & \text{otherwise.} \end{cases}$$

Algorithm DP

```
Initialisation:
01  for each combination of $i, i_1, j, k$ satisfying $1\leqslant i\leqslant i_1\leqslant j\leqslant n$, and $\lceil(j-i_1)/j\rceil+1\leqslant k\leqslant j-i_1+1$ do
02      $g(i,i_1,j,k) = \begin{cases} f_1(j), & \text{if } i=k=1 \text{ and } i_1=j; \\ \infty, & \text{otherwise.}\end{cases}$
Recursion:
03  for each combination of $i$, $i_1$, $j$, $k$ satisfying $2\leqslant i\leqslant i_1\leqslant j\leqslant n$, and $\lceil(j-i_1)/j\rceil+1\leqslant k\leqslant j-i_1+1$ do
04      Set $z=\infty$;
05      Case $k=1$ (Figure 4):
06      for each combination of $i', i'_1, k'$ satisfying $1\leqslant i'\leqslant i'_1\leqslant i-1$, and $\lceil(i-i'_1-1)/(i-1)\rceil+1\leqslant k'\leqslant i-i'_1$ do
07          if $C_{1,j}-C_{1,i'_1}>k's+p_{2,[i':i-1]}$
08              then $temp=f_2(i', i'_1, i, k', j)$;
09          else $temp=\infty$;
10          $z=\min\{z, temp\}$.
11      Case $k>1$ (Figure 5):
12      for each $j'$ from $i_1+k-2$ up to $j-1$ do
13          if $C_{1,j}-C_{1,i_1}\leqslant(k-1)s+p_{2,[i:j']}$
14              then $temp=f_3(i, i_1, j', k, j)$;
15          else $temp=\infty$;
16          $z=\min\{z, temp\}$.
17      $g(i,i_1,j,k)=z$.
```

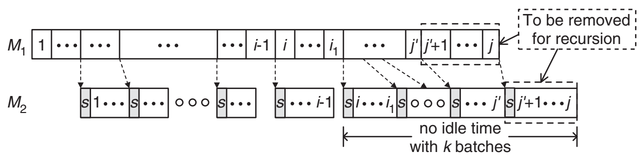

Figure 5 Forward recursion of the case $k>1$.

Goal: Find min $\{g(i, i_1, n, k) | 1 \leqslant i \leqslant i_1 \leqslant n, \lceil (n-i_1)/n \rceil + 1 \leqslant k \leqslant n-i_1+1\}$.

If Algorithm *DP* is deployed for the performance metric of total completion time, Function $f_1(j)$, $f_2(i', i'_1, i, k', j)$ and $f_3(i, i_1, j', k, j)$ are denoted as follows.

In line 02, $f_1(j) = j(C_{1,j} + s + p_{2,[1:j]})$.

In line 08, $f_2(i', i'_1, i, k', j) = g(i', i'_1, i-1, k') + (j-i+1)(C_{1,j} + s + p_{2,[i:j]})$.

In line 14, $f_3(i, i_1, j', k, j) = g(i, i_1, j', k-1) + (j-j')(C_{1,i_1} + ks + p_{2,[i:j]})$.

For the objective function of maximum lateness, we define

$$f_1(j) = C_{1,j} + s + p_{2,[1:j]} - d_{[1:j]},$$
$$f_2(i', i'_1, i, k', j) = \max\{g(i', i'_1, i-1, k'), C_{1,j} + s + p_{2,[i:j]} - d_{[i:j]}\}, \text{ and}$$
$$f_3(i, i_1, j', k, j) = \max\{g(i, i_1, j', k-1), (C_{1,i_1} + ks + p_{2,[i:j]} - d_{[j'+1:j]})\}.$$

For the objective function of total tardiness, we have

$$f_1(j) = \sum_{h=1}^{j} T_h(C_{1,j} + s + p_{2,[1:j]}),$$
$$f_2(i', i'_1, i, k', j) = g(i', i'_1, i-1, k') + \sum_{h=i}^{j} T_h(C_{1,j} + s + p_{2,[i:j]}), \text{ and}$$
$$f_3(i, i_1, j', k, j) = g(i, i_1, j', k-1) + \sum_{h=j'+1}^{j} T_h(C_{1,i_1} + ks + p_{2,[i:j]}).$$

As for the number of tardy jobs, we denote

$$f_1(j) = \sum_{h=1}^{j} U_h(C_{1,j} + s + p_{2,[1:j]}),$$
$$f_2(i', i'_1, i, k', j) = g(i', i'_1, i-1, k') + \sum_{h=i}^{j} U_h(C_{1,j} + s + p_{2,[i:j]}), \text{ and}$$
$$f_3(i, i_1, j', k, j) = g(i, i_1, j', k-1) + \sum_{h=j'+1}^{j} U_h(C_{1,i_1} + ks + p_{2,[i:j]}).$$

As shown in Figure 4, the if-condition in line 07 is utilised to examine whether a partial schedule in state $(i, j, j, 1)$ can be constructed by attaching jobs $i, \ldots, j$, which constitute a single batch on M_2, to the partial schedule in state $(i', i'_1, i-1, k')$ with value $g(i', i'_1, i-1, k')$. Figure 5 illustrates that the if-condition in line 13 is used to check whether a partial schedule in state (i, i_1, j, k) can be constructed by merging the batch of jobs $j'+1, \ldots, j$ with the partial schedule achieving $g(i, i_1, j', k-1)$ in state $(i, i_1, j', k-1)$. The optimal objective value is $\min\{g(i, i_1, n, k) | 1 \leqslant i \leqslant i_1 \leqslant n, \lceil (n-i_1)/n \rceil + 1 \leqslant k \leqslant n-i_1+1\}$ and the corresponding optimal schedule can be obtained by backtracking.

The recursion part of Algorithm DP consists of two cases: (1) For the case $k=1$ (also implying $i_1=j$), there are $O(n^2)$ states $(i, j, j, 1)$, each of which requires $O(n^3)$ operations; (2) For the case $k>1$, there are $O(n^4)$ states (i, i_1, j, k), each of which needs $O(n)$ operations. Note that for any considered performance measure, function $f_1(j)$, $f_2(i', i'_1, i, k', j)$ and $f_3(i, i_1, j', k, j)$ can be calculated in constant time or by a straightforward pre-processing procedure. The goal step requires $O(n^3)$ comparisons, each of which takes constant time. Hence, the running time of algorithm DP is $O(n^5)$.

Since the first and second phases respectively require $O(mn)$ and $O(n^5)$ times, the running time of the proposed two-phase algorithm is $O(mn+n^5)$.

Theorem 1 *Problem* $(m+1)MAF|m\delta \rightarrow \beta$, *s-batch, fixed_ seq*$|\gamma$, *where* $\gamma = \sum C_j$, $L_{\max}$, $\sum T_j$ *or* $\sum U_j$, *can be solved in* $O(mn+n^5)$ *time.*

Example Consider the following instance with $m=2$ and $n=4$: $(p_{a,1}, p_{b,1}, p_{2,1}) = (2, 1, 1)$, $(p_{a,2}, p_{b,2}, p_{2,2}) = (1, 3, 3)$, $(p_{a,3}, p_{b,3}, p_{2,3}) = (4, 2, 2)$, $(p_{a,4}, p_{b,4}, p_{2,4}) = (1, 3, 1)$, and $s=1$. The two-phase algorithm is demonstrated for the objective function of total completion time as follows:

Phase 1: (Problem-transformation procedure)

$$C_{1,1} = \max\{2, 1\} = 2;$$
$$C_{1,2} = \max\{2+1, 1+3\} = 4;$$
$$C_{1,3} = \max\{2+1+4, 1+3+2\} = 7;$$
$$C_{1,4} = \max\{2+1+4+1, 1+3+2+3\} = 9.$$

Phase 2: (Algorithm DP-Batch)

Initialisation

$$g(1,1,1,1) = 1 \times (2+1+1) = 4;$$
$$g(1,2,2,1) = 2 \times (4+1+4) = 18;$$
$$g(1,3,3,1) = 3 \times (7+1+6) = 42;$$
$$g(1,4,4,1) = 4 \times (9+1+7) = 68;$$

For other values of i, i_1, j, k, we denote $g(i, i_1, j, k) = \infty$.

Recursion

$$\begin{aligned}
g(2,2,2,1) &= \infty;\\
g(2,2,3,2) &= g(2,2,2,1) + (3-2)\\
&\quad \times \left(C_{1,2} + 1 \times 1 + p_{2,[2:3]}\right) = \infty;\\
g(2,2,4,2) &= \min\{\infty, g(2,2,3,1) + (4-3)\\
&\quad \times \left(C_{1,2} + 2 \times 1 + p_{2,[2:4]}\right)\} = \infty;\\
g(2,2,4,3) &= g(2,2,3,3) + (4-3)\\
&\quad \times \left(C_{1,2} + 3 \times 1 + p_{2,[2:4]}\right) = \infty;\\
g(2,3,3,1) &= g(1,1,1,1) + (3-2+1)(7+1+5)\\
&= 30;\\
g(2,3,4,2) &= g(2,3,3,1) + (4-3)\\
&\quad \times (7+2+6) = 30 + 15 = 45;\\
g(2,4,4,1) &= g(1,1,1,1) + (4-2+1)\\
&\quad \times (9+1+6) = 4 + 48 = 52;\\
g(3,3,3,1) &= \infty;\\
g(3,3,4,2) &= g(3,3,3,1) + (4-3)\\
&\quad \times \left(C_{1,3} + 2 \times 1 + p_{2,[3:4]}\right) = \infty;\\
g(3,4,4,1) &= \min\{g(1,1,2,2) + (4-3+1)\\
&\quad \times (9+1+3), \infty, g(2,2,2,1)\\
&\quad + (4-3+1)(9+1+3)\} = \infty;\\
g(4,4,4,1) &= \infty.
\end{aligned}$$

Goal

$$\begin{aligned}
&\min\{g(i, i_1, 4, k) | 1 \leqslant i \leqslant i_1 \leqslant 4, \left\lceil \frac{4-i_1}{4} \right\rceil\\
&\quad + 1 \leqslant k \leqslant 4 - i_1 + 1\}\\
&= \min\{g(1,4,4,1), g(2,3,4,2)\\
&\quad g(2,4,4,1)\} = 45.
\end{aligned}$$

The optimal schedule (2, 3, 4, 2) can be constructed by backtracking the recursion, as demonstrated in Figure 6.

5. Concluding remarks

A two-stage assembly-type flowshop batch scheduling problem with a fixed job sequence has been addressed in this study. For the minimisation of total completion time, maximum lateness, total tardiness or number of tardy jobs, this study designed an $O(mn + n^5)$-time two-phase algorithm, where m is the number of parallel dedicated machines arranged at stage 1 and n is the number of jobs. The running time will be $O(n^5)$ if the number of dedicated machines m is not a part of the input. Besides, problem $F2|\delta \rightarrow \beta$, *s-batch*, *fixed_seq*$|\gamma$ was solved in $O(n^5)$ time in this study. Furthermore, the developed algorithm can be easily generalised for the weighted counterparts.

Future study may be conducted on the parallel-batch (max-batch) model, that is, $(m+1)\text{MAF}|m\delta \rightarrow \beta$, *p-batch*, *fixed_seq*$|\gamma$. Furthermore, we can address the reverse model which is a two-stage $(1+m)$-machine flowshop dismantling and refurbishing products. In the reverse model, stage 1 is equipped with a dismantling line which disassembles products in batches. There are m parallel dedicated machines arranged at stage 2 for processing the m individual component parts generated from the dismantled products. This reverse model can also be studied with the sum-batch or max-batch model.

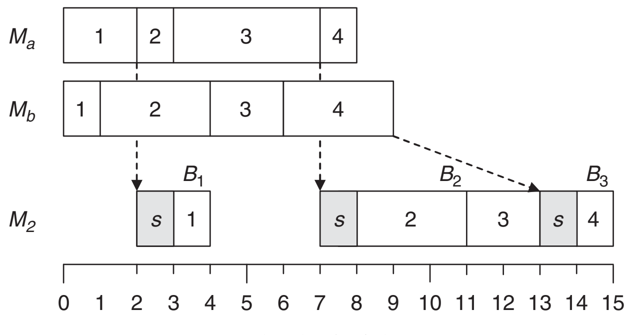

Figure 6 Optimal schedule (2, 3, 4, 2).

Acknowledgements—The authors are grateful to the anonymous reviewers for their constructive comments that have improved an earlier version of this paper.

References

Allahverdi A, Gupta JND and Aldowaisan T (1999). A review of scheduling research involving setup considerations. *Omega* **27**: 219–239.

Cheng TCE and Wang G (1999). Scheduling the fabrication and assembly of components in a two-machine flowshop. *IIE Trans* **31**: 135–143.

Cheng TCE, Gupta JND and Wang G (2000a). A review of flowshop scheduling research with setup times. *Prod Opns Mngt* **9**: 262–282.

Cheng TCE, Lin BMT and Tian YM (2009). Scheduling of a two-stage differentiation flowshop to minimise weighted sum of machine completion times. *Comput Opns Res* **36**: 3031–3040.

Cheng TCE, Lin BMT and Toker A (2000b). Makespan minimization in the two-machine flowshop batch scheduling problem. *Nav Res Log* **47**: 128–144.

Herrmann JW and Lee CY (1992). *Three-machine look-ahead scheduling problems*. Research Report No. 92–93, Department of Industrial Engineering, University of Florida, FL.

Hwang FJ (2011). *Scheduling problems subject to fixed job sequences*. PhD Dissertation, Institute of Information Management, National Chiao Tung University, Taiwan.

Hwang FJ and Lin BMT (2011). Coupled-task scheduling on a single machine subject to a fixed-job-sequence. *Comput Indust Eng* **60**: 690–698.

Hwang FJ, Kovalyov MY and Lin BMT (2010). Minimization of total completion time in flowshop scheduling subject to fixed job sequences. In: *Proceedings of 12th International Workshop on Project Management and Scheduling*, Tours, France, pp 249–252.

Johnson SM (1954). Optimal two- and three-stage production schedules with setup times included. *Nav Res Log Q* **1**: 61–68.

Lee CY, Cheng TCE and Lin BMT (1993). Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem. *Mngt Sci* **39**: 616–625.

Lin BMT and Cheng TCE (2005). Two-machine flowshop batching and scheduling. *Ann Opns Res* **133**: 149–161.

Lin BMT and Cheng TCE (2006). Two-machine flowshop scheduling with conditional deteriorating second operations. *Int Trans Opl Res* **13**: 91–98.

Lin BMT and Cheng TCE (2011). Scheduling with centralized and decentralized batching policies in concurrent open shops. *Nav Res Log* **58**: 17–27.

Lin BMT and Hwang FJ (2011). Total completion time minimization in a 2-stage differentiation flowshop with fixed sequences per job type. *Inform Process Lett* **111**: 208–212.

Lin BMT, Cheng TCE and Chou ASC (2007). Scheduling in an assembly-type production chain with batch transfer. *Omega—Int J Mngt Sci* **35**: 143–151.

Ng CT and Kovalyov MY (2007). Batching and scheduling in a multi-machine flow shop. *J Scheduling* **10**: 353–364.

Potts CN and Kovalyov MY (2000). Scheduling with batching: A review. *Eur J Opl Res* **120**: 228–249.

Potts CN and Van Wassenhove LN (1992). Integrating scheduling with batching and lot-sizing: A review of algorithms and complexity. *J Opl Res Soc* **43**: 395–406.

Potts CN and Whitehead JD (2007). Heuristics for a coupled-operation scheduling problem. *J Opl Res Soc* **58**: 1375–1388.

Shafransky YM and Strusevich VA (1998). The open shop scheduling problem with a given sequence of jobs on one machine. *Nav Res Log* **45**: 705–731.

Received February 2011;
accepted June 2011 after one revision