Available online at www.sciencedirect.com
SCIENCE DIRECT
Omega 35 (2007) 143–151
omega
The International Journal of Management Science
www.elsevier.com/locate/omega

Scheduling in an assembly-type production chain with batch transfer☆

B.M.T. Lin[a,*], T.C.E. Cheng[b], A.S.C. Chou[c]

[a]*Department of Information and Finance Management, Institute of Information Management, National Chiao Tung University, Hsinchu, 300 Taiwan*
[b]*Department of Logistics, The Hong Kong Polytechnic University, Kowloon, Hong Kong*
[c]*Department of Information Management, Ming Chuan University, Taoyuan, 333 Taiwan*

Received 24 September 2004; accepted 29 April 2005
Available online 28 June 2005

Abstract

This paper addresses a three-machine assembly-type flowshop scheduling problem, which frequently arises from manufacturing process management as well as from supply chain management. Machines one and two are arranged in parallel for producing component parts individually, and machine three is an assembly line arranged as the second stage of a flowshop for processing the component parts in batches. Whenever a batch is formed on the second-stage machine, a constant setup time is required. The objective is to minimize the makespan. In this study we establish the strong NP-hardness of the problem for the case where all the jobs have the same processing time on the second-stage machine. We then explore a useful property, based upon which a special case can be optimally solved in polynomial time. We also study several heuristic algorithms to generate quality approximate solutions for the general problem. Computational experiments are conducted to evaluate the effectiveness of the algorithms.

Keywords: Two-stage flowshop; Batch scheduling; Makespan; NP-hardness; Heuristics

1. Introduction

In this paper we study an assembly-type production scheduling problem, which can be used to model the coordination of production scheduling between cooperative parties in a supply chain. Consider a set of jobs (or products) to be processed from time zero onwards in a two-stage flowshop with three machines (or party firms in a supply chain). In the machine configuration, the first stage has two parallel machines whose outputs, i.e., components or parts, will be transferred to the second-stage machine, which is dedicated to assembly operations. Each job has three specific operations to be performed on the three machines, respectively. Each machine can process at most one operation at a time. No preemption is allowed. Operations on the second-stage machine are processed in batches and a constant setup time is needed whenever a batch is formed. The setup is non-anticipatory, i.e., a setup can commence only when all the parts of the jobs in the same batch are transferred to and available on the assembly machine. Batch availability is assumed for the batch process, i.e., a job is

☆ The first author was partially supported by the National Science Council under Grant NSC-93-2416-H-009-026. The second author was supported in part by The Hong Kong Polytechnic University under a grant from the *Area of Strategic Development in China Business Services*.

* Corresponding author. Tel.: +886 3 5131472; fax: +886 5 729915.
E-mail address: bmtlin@mail.nctu.edu.tw (B.M.T. Lin).

0305-0483/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.omega.2005.04.004

finished when the batch it belongs to is completed as a whole. The objective is to sequence, as well as to group, the jobs to minimize the makespan, i.e., the maximum completion time of all the jobs. Notice that centralized decision making is assumed. That is, the sequencing and batching policies are determined for the assembly machine and applied to the other two machines.

The problem under consideration is related to two well-studied scheduling problems, namely the hybrid flowshop scheduling problem and the batch scheduling problem. Johnson [1] first introduced the flowshop scheduling model and proposed a solution algorithm to minimize the makespan in a two-machine environment. In the past few decades, this seminal work has inspired numerous research endeavors in the scheduling literature [2,3]. As a generalization of Johnson's two-machine flowshop, the three-machine assembly flowshop problem motivated by the manufacture of fire engines was studied by Lee et al. [4]. Like the three-machine flowshop problem [5], the problem to minimize the makespan in an assembly flowshop becomes computationally intractable. This kind of production setting is not only common in individual manufacturing organizations but also prevalent in supply chains where a manufacturer receives parts or materials from its upstream suppliers for final assembly or packaging. The incorporation of batch considerations into the scheduling model is motivated by the observation that components or parts are usually delivered to a downstream party in batches, such as in full truck loads (FTL). In supply chain management, either centralized or decentralized decision making can be assumed to reflect real situations [6]. In the problem under consideration, we assume that the assembly organization is dominant in the industry and therefore decides the production policies for optimizing the objectives. Also, we assume that the two-part suppliers begin their processing at the same time and their production processes dedicated to the n jobs (orders) must be continuous and cannot be interrupted by any other orders or requests. For heuristic algorithms for the three-machine assembly-type production scheduling, the reader is referred to Sun et al. [7].

Batching is one of the major characteristics of the studied problem. The major advantage of batching is the achievement of gains in operational efficiency that results from setup reductions. Over the past few decades, combining scheduling with batching has received significant research attention. Interest in batch scheduling is due to its relevance to real-world manufacturing and its theoretical challenges. Potts and Van Wassenhove [8], and Webster and Baker [9] have reviewed different batching models. In three recent survey papers by Allahverdi et al. [10], Cheng et al. [11], and Potts and Kovalyov [12], concise and comprehensive reviews on scheduling problems with batching and setup times/costs were presented. Amongst the different models, Lee et al. [13] studied the so-called "burn-in" operations in the semiconductor industry. In the burn-in model, the processing time of a batch is defined as the longest processing time of the jobs contained in the batch. Ahmadi et al. [14] considered a batch-scheduling problem in a two-machine flowshop, where the processing time of a batch is constant regardless of the number and type of jobs it contains. The batching model considered in this study was previously studied by Albers and Brucker [15], Coffman et al. [16] and Santos and Magazine [17]. In this model the jobs assigned in the same batch require a common setup and their processing is continuous on the machine. Therefore, the processing time of a batch is the setup time plus the total processing times of the jobs belonging to the batch. Following the continuous batching model, Cheng and Wang [18] studied a two-machine flowshop in which the operations on the first machine are processed individually while the operations on the second machine are processed in batches. They showed that the problem is NP-hard and identified some polynomially solvable cases. Cheng et al. [19] considered the same configuration except that both machines process the jobs in batches. They presented strong NP-hardness proofs and developed efficient algorithms for several special cases. Glass et al. [20] studied a scheduling model similar to that of Cheng et al. [19] with anticipatory machine-dependent setup times. Theoretically, the problem we study in this paper concerns a combination of the models presented by Lee et al. [4] and Cheng and Wang [18]. The general case with multiple machines in stage one has been studied by Kovalyov et al. [21]. They proposed a lower bound and a heuristic algorithm, and presented a performance ratio analysis of the heuristic.

This paper is organized into six sections. In Section 2 we present the notation used in this paper and give an example to illustrate the problem definition. In Section 3 we show the strong NP-hardness of the problem. Section 4 is dedicated to studying a special case that is polynomially solvable. In Section 5 we investigate several heuristics for finding approximate solutions. The results of the computational experiments conducted to evaluate the performance of proposed algorithms are discussed. Finally, we give some concluding remarks in Section 6.

2. Notation and example

In this section we introduce the notation that will be used in this paper. Also, we give a numerical example to illustrate the problem definition.

Notation:
$N = \{1, 2, \ldots n\}$ job set to be processed
M_a, M_b: two first-stage machines
M_2: second-stage machine
p_{ia}: processing time of job i on machine M_a
p_{ib}: processing time of job i on machine M_b
p_{i2}: processing time of job i on machine M_2
s: batch setup time
S: schedule for the job set N
$Z(S)$: makespan of schedule S
$Z^*(N)$: optimal makespan for job set N

The problem under consideration can be formulated as follows: there is a set of jobs $N = \{1, 2, \ldots, n\}$ simultaneously available at time zero for processing in a two-stage flowshop, in which two independent dedicated machines M_a and M_b are deployed in stage one and one assembly machine M_2 in stage 2. Each job i in N consists of three operations that are processed on the three machines. The processing times are p_{ia}, p_{ib}, and p_{i2}, respectively. For job i, when its two operations on the stage-one machines are completed, the two parts will be transferred to the stage-two machine for assembling. While the stage-one machines process the jobs individually, the stage-two machine processes the jobs in batches with a constant setup time s whenever a batch is formed. The problem seeks to sequence, as well as to group, the jobs to minimize the makespan, i.e., the maximum completion time amongst the jobs.

To simplify presentation and to denote the problem under study, we will use the three-field notation introduced by Lawler et al. [22] with some extensions. In this extended notation the problem will be denoted by $3\text{MAF}|(\delta, \delta) \to \beta|C_{\max}$, where MAF signifies "machine assembly flowshop". The second field $(\delta, \delta) \to \beta$ indicates that the manufacturing environment contains two discrete processors at stage one and a batch processor at stage two. Next, we give an example as an illustration of the problem definition. A set of six jobs is given as follows:

Job	1	2	3	4	5	6
p_{ia}	1	2	2	5	3	1
p_{ib}	2	3	3	4	1	4
p_{i2}	3	4	1	2	2	2

The batch setup time is 1. Let $S_1 = \{\{1, 2\}, \{3, 4\}, \{5, 6\}\}$ and $S_2 = \{\{2, 4, 6\}, \{1, 3, 5\}\}$ be two schedules for the given job set. Indices enclosed within inner braces are jobs grouped in the same batch. Schedules S_1 and S_2 have three and two batches, respectively. The Gantt charts of the two schedules are shown in Fig. 1 . Schedule S_1 has a smaller makespan than schedule S_2, although it has one more setup.

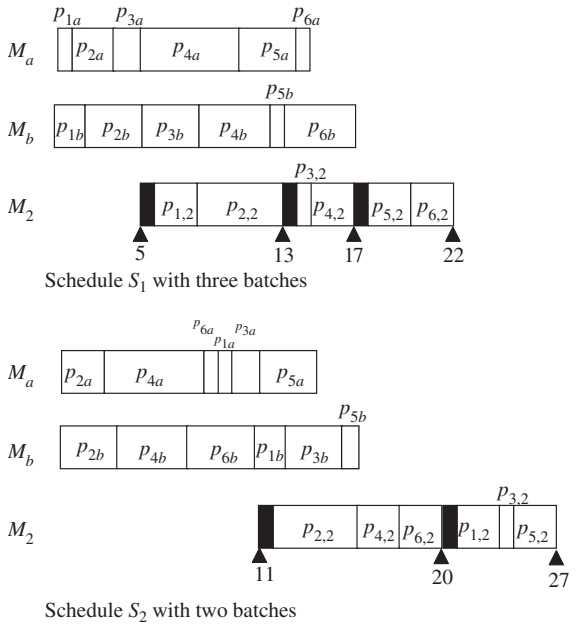

Fig. 1. Two example schedules.

3. NP-hardness

The $F2|\delta \to \beta|C_{\max}$ problem, where a discrete processor and a batch processor are arranged into a two-machine flowshop, is known to be NP-hard [18]. Therefore, the $3\text{MAF}|(\delta, \delta) \to \beta|C_{\max}$ problem, as a generalization of $F2|\delta \to \beta|C_{\max}$, is naturally NP-hard, even if either machine M_a or machine M_b is ignored. The special case where all the jobs have the same processing time on the second-stage machine is polynomially solvable for both the $F2|\delta \to \beta|C_{\max}$ problem [18] and the $3\text{MAF}\|C_{\max}$ problem [4]. In this section we shall show that the special case of the later problem $3\text{MAF}|(\delta, \delta) \to \beta$, $p_{i2} = p|C_{\max}$ becomes strongly NP-hard when batching is considered. The proof is based upon a reduction from the 3-Partition problem, which is known to be strongly NP-complete [23].

3-Partition. A number $m \in Z^+$, a bound $E \in Z^+$ and a finite set A of $3m$ non-negative integers $\{x_1, x_2, \ldots, x_{3m}\}$ with $E/4 < x_i < E/2$ for all i and $\sum_{i=1}^{3m} x_i = mE$ are given. Can A be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that $\sum_{x_i \in A_j} x_i = E$ for $1 \leqslant j \leqslant m$?

Theorem 1. *The decision version of the* $3\text{MAF}|(\delta, \delta) \to \beta$, $p_{i2} = p|C_{\max}$ *problem is strongly* NP*-complete.*

Proof. The decision version of the special case $3\text{MAF}|(\delta, \delta) \to \beta$, $p_{i2} = p|C_{\max}$ is clearly in NP. Without loss of generality, we assume that for the given instance of 3-Partition, $E > 3m + 3$ and for all i, x_i is a multiple of 3. If this is not the case, we can adjust the values by multiplying them by 3. An instance of $3\text{MAF}|(\delta, \delta) \to \beta$, $p_{i2} = p|C_{\max}$ with $3m + 3$ jobs is constructed as follows:

$$
\begin{aligned}
&N = \{1, 2, \ldots, 3m + 3\};\\
&p_{ia} = (E + x_i)\omega, \quad p_{ib} = (2E - 2x_i)\omega, \quad \text{and}\\
&p_{i2} = 4\omega E/3 - 1, \quad \text{for } i = 1, 2, \ldots, 3m,\\
&p_{ia} = 0,\ p_{ib} = 0, \quad \text{and} \quad p_{i2} = 4\omega E/3 - 1\\
&\quad \text{for } i = 3m + 1, 3m + 2, 3m + 3,
\end{aligned}
$$

where $\omega = 10E^2$, and setup time $s = 3$.

Recall that E is divisible by 3 because any element in A is a multiple of 3. We claim that a partition of the given instance of 3-Partition exists if and only if there is an optimal schedule for the instance of the 3MAF$|(\delta, \delta) \to \beta, p_{i2} = p|C_{\max}$ problem with makespan no greater than $4(m+1)\omega E$. For details of the remaining part of the proof, the reader is referred to the appendix. □

The theorem indicates that the 3MAF$|(\delta, \delta) \to \beta|C_{\max}$ problem remains intractable even if all the jobs have the same processing time on the batch processor. As a consequence, it is very unlikely that efficient algorithms can be devised to optimally solve the general problem. In the following sections we first investigate some further restricted special cases that can be solved in polynomial time. Based on these studied special cases, we then propose heuristics that can produce quality approximate solutions to the general problem.

4. Polynomially solvable case

In this section we deal with a property concerning the polynomial solvability of the problem. Based on the derived results, we further explore a property that is useful for computing a lower bound for optimal solutions. As discussed before, to optimally compose a solution to the 3MAF$|(\delta, \delta) \to \beta|C_{\max}$ problem we need to take batching and sequencing issues into account simultaneously. In this section we assume that for an input instance, a job sequence is given and fixed. Then, we show that an optimal batching scheme is attainable in polynomial time.

For simplicity of presentation, we assume that the jobs follow the sequence $1, 2, \ldots, n$ for processing. For any two schedules S_1 and S_2, i.e., they are batch compositions of the first i jobs, $1, 2, \ldots, i$, that have the same completion time on the stage-one machines. If $Z(S_1) \leqslant Z(S_2)$, then

$$Z(S_1 \cup B_k) \leqslant Z(S_2 \cup B_k),$$

where B_k is the batch consisting of the remaining jobs, $i+1, i+2, \ldots, n$, must hold. That is, if the last batch, denoted by B_k, is confined to containing jobs $i+1, i+2, \ldots, n$ only, then to find an optimal schedule for N, we can first determine an optimal schedule for jobs $1, 2, \ldots, i$, and then append the last batch B_k to this prefix schedule. The above observations lead to the development of a recursive formulation for an optimal composition scheme. Let $F(i)$ be the optimum completion time among all the schedules for the first i jobs. We have the following recursive algorithm.

Algorithm DP

Initial conditions: $F(0) = 0$; $F(i) = \infty$, if $i \neq 0$.

Recursive formula:

$$F(i) = \min_{\alpha=1,2,\ldots,i} \left\{ \max \left\{ F(i-\alpha), \sum_{j=1}^{i} P_{ja}, \sum_{j=1}^{i} P_{jb} \right\} + s + \sum_{j=i-\alpha+1}^{i} P_{j2} \right\}.$$

Goal : $Z^*(N) = F(n)$.

The algorithm is easy to justify by the above observations. In the recursive formula, variable α denotes the number of jobs to be included in the last batch. The recursive function $F()$ has $O(n)$ states, each of which takes $O(n)$ time to determine its optimal value. Therefore, the time complexity of *Algorithm DP* is $O(n^2)$. There holds the following result.

Theorem 2. *For a fixed job sequence for the* 3MAF$|(\delta, \delta) \to \beta|C_{\max}$ *problem, an optimal batch composition can be determined in* $O(n^2)$ *time.*

Next, we consider a special case that satisfies the following condition $C1$: *For any jobs* i *and* j *in* N, $p_{ia} \leqslant p_{ja} \Leftrightarrow p_{ib} \leqslant pjb \Leftrightarrow p_{i2} \geqslant p_{j2}$.

Lemma 1. *For the special case satisfying condition* $C1$, *there exists an optimal schedule* S *in which for any two jobs* i *and* j, *if* $p_{ia} \leqslant p_{ja}$, $p_{ib} \leqslant p_{jb}$ *and* $p_{i2} \geqslant p_{j2}$, *then either jobs* i *and* j *are in the same batch or the batch containing job* i *precedes the batch containing job* j.

Proof. Assume that there is an optimal schedule where jobs i and j are not in the same batch and the batch containing job j precedes the batch containing job i. It is easy to see that swapping the positions of the two jobs will not increase the makespan. Continuing the job interchange arguments, if necessary, will finally lead to an optimal schedule possessing the characteristics specified in the lemma. □

This result indicates that for the special case, the jobs can be arranged according to the relations specified in the lemma. Combining Theorem 2 and Lemma 1, we have the following result.

Corollary 1. *The special case satisfying condition* $C1$ *can be optimally solved in* $O(n^2)$ *time.*

The above results not only provide exact solutions for a special case but also shed light on the development of an estimate of the optimal solution for the general 3MAF$|(\delta, \delta) \to \beta|C_{\max}$ problem. As the general problem is strongly NP-hard, it is very unlikely to come up with a fast algorithm to produce optimal solutions for large-scale problems. Therefore, estimations of the optimal solutions are needed. Next,

we derive a lower bound for the problem. The idea follows from Cheng et al. [18].

For job set N, we first remove the associations among p_{ia}, p_{ib} and p_{i2} for all jobs i, and then create an ideal job set N' such that job i' consists of parameters p'_{ia}, p'_{ib} and p'_{i2}, where p'_{ia} is the ith smallest value in $\{p_{1a}, p_{2a}, \ldots, p_{na}\}$, p'_{ib} is the ith smallest value in $\{p_{1b}, p_{2b}, \ldots, p_{nb}\}$ and p'_{i2} is the ith largest value in $\{p_{12}, p_{22}, \ldots, p_{n2}\}$. Revising the data set used in Section 2 in this manner, we have the derived instance as follows:

Job	$1'$	$2'$	$3'$	$4'$	$5'$	$6'$
p'_{ia}	1	1	2	2	3	5
p'_{ib}	1	2	3	3	4	4
p'_{i2}	4	3	2	2	2	1

It is clear that the derived instance N' satisfies condition $C1$ and thus an optimal schedule for N' is attainable in $O(n^2)$ time. In the following lemma, a relationship is established between the solutions for sets N and N'.

Theorem 3. *For the job set N and the derived job set* N', $Z^*(N') \leqslant Z^*(N)$.

Proof. Given schedule S optimal for set N, we have the following transformation process to derive schedule S' such that $Z(S') \leqslant Z(S) = Z^*(N)$. For any two consecutive jobs i and j in schedule S, if job i precedes job j and $p_{ja} < p_{ia}$, we swap operations p_{ja} and p_{ia} and retain p_{ib}, p_{jb}, p_{i2} and p_{j2} in their original positions. The makespan will not increase by this swapping operation. Continue the swapping process until there are no such jobs as i and j. In the derived schedule, all the operations on machine M_a are arranged in non-decreasing order of their processing times, and the makespan is no greater than that of schedule S. With this derived new schedule, rearranging the processing times on M_b in non-decreasing order will not increase the makespan either. Then, we rearrange the processing times on machine M_2 in a non-increasing order. Similarly, the makespan will not increase due to the transformation. After the above transformations, we obtain schedule S' in which the makespan $Z(S')$ is no greater than $Z(S)$ and the jobs satisfy the definition of the job set N'. By the definition of $Z^*(N')$, we have $Z^*(N') \leqslant Z(S')$. □

Summarizing the above results, we have come up with a two-step procedure for deriving a lower bound on the optimal solution for the original job set. The first step is to transform the job set N into N', and the second step applies *Algorithm DP* to find an optimal schedule for N'. The derived solution, by Theorem 3, is a lower bound on the optimal solution value of set N. Therefore, a lower bound can be computed in $O(n^2)$ time.

5. Heuristic algorithms and computational experiments

The strong NP-hardness result presented in Section 2 hints that it is very unlikely to design efficient algorithms to optimally solve the $3\text{MAF}|(\delta, \delta) \rightarrow \beta|C_{\max}$ problem. Furthermore, it is difficult to devise a branching tree for branch-and- bound algorithms due to the fact that the scheduling decisions consist of deciding jointly on how to batch the jobs and how to sequence the batches. Similarly, the design of heuristic and meta-heuristic algorithms may be based upon the theme of considering the two decisions at the same time. The results of Theorem 2, however, suggest an alternative—determining a job sequence and applying *Algorithm DP* to the job sequence to optimally assign the jobs into batches. In this section we follow this line of design and develop four heuristic algorithms to obtain approximate solutions in a reasonable time.

In our study we divide the problem-solving strategy into two phases, namely job sequencing and batch composition. The first three sequencing procedures H_1, H_2 and H_3 were proposed by Lee et al. [4] for obtaining job sequences as approximate solutions.

Algorithm H_1

Step 1: Let $p_{i1} = \max\{p_{ia}, p_{ib}\}$, for $i = 1, 2, \ldots, n$.

Step 2: Apply Johnson's algorithm to the job instance with job i defined by p_{i1} and p_{i2}.

Step 3: Apply *Algorithm DP* to generate an optimal batching policy for the sequence determined in Step 2.

Algorithm H_2

Step 1: If $\sum_{i=1,\ldots,n} p_{ia} \geqslant \sum_{i=1,\ldots,n} p_{ib}$, then $p_{i1} = p_{ia}$; otherwise, $p_{i1} = p_{ib}$.

Steps 2 *and* 3: Same as *Algorithm* H_1.

Algorithm H_3

Step 1: Let $p_{i1} = (p_{ia} + p_{ib})/2$.

Steps 2 *and* 3: Same as *Algorithm* H_1.

We propose the following heuristic for the problem.

Algorithm H_4

Step 1: Let $p_{i1} = (p_{ia} + p_{ib})/p_{i2}$ for $i = 1, 2, \ldots, n$.

Step 2: Arrange the jobs in non-decreasing order of p_{i1}.

Step 3: Apply *Algorithm DP* to the sequence determined in Step 2.

As for the time complexity, all of the above algorithms take $O(n \log n)$ time to determine a job sequence and $O(n^2)$ time to apply *Algorithm DP*. Therefore, the algorithms all have a time complexity of $O(n^2)$.

To study the effectiveness of the above algorithms, we conducted a series of computational experiments. The programs were coded in Visual C++ 6.0 and run on a personal computer. The lower-bound concept was deployed to serve as the baseline for comparisons. Let Z_H be the schedule derived by algorithm H and Z_{lb} the schedule determined by the lower bound introduced in Section 4. The relative error ratio of algorithm H is defined as $(Z_H - Z_{lb})/Z_{lb} \times 100\%$.

As regards the data set preparation, the processing times p_{ia}, p_{ib} and p_{i2} were randomly drawn from the interval

Table 1
Average relative error ratios (%) of the heuristics

Number of jobs	Setup time																			
	$s = 10$				$s = 30$				$s = 50$				$s = 150$				$s = 500$			
	H_1	H_2	H_3	H_4	H_1	H_2	H_3	H_4	H_1	H_2	H_3	H_4	H_1	H_2	H_3	H_4	H_1	H_2	H_3	H_4
10	3.54	5.56	3.52	3.41	3.70	4.42	3.76	3.32	3.63	4.61	3.87	3.34	3.76	4.32	3.69	3.07	1.69	1.83	1.82	1.61
50	2.11	4.52	2.25	2.04	2.03	4.47	2.27	2.02	2.17	4.57	2.39	2.05	3.01	4.86	2.95	2.74	3.12	4.83	3.17	2.79
100	1.41	3.33	1.33	1.04	1.31	3.13	1.36	1.07	1.44	3.37	1.51	1.08	2.65	4.79	2.59	2.09	3.05	5.40	3.08	2.64
150	1.16	2.92	0.93	0.86	1.06	2.88	0.97	0.87	1.13	3.21	1.10	0.93	2.10	4.24	2.13	1.98	2.86	5.10	2.98	2.49
200	0.92	2.67	0.89	0.73	0.86	2.56	0.88	0.72	0.91	2.70	0.94	0.73	1.80	3.47	1.79	1.63	2.79	5.11	2.83	2.34
250	0.59	1.75	0.56	0.41	0.59	1.74	0.60	0.40	0.53	1.71	0.61	0.41	1.54	3.52	1.76	1.65	2.57	4.49	2.58	2.10
300	0.64	1.87	0.58	0.42	0.49	1.81	0.52	0.43	0.52	1.84	0.54	0.43	1.39	3.41	1.48	1.10	2.40	4.75	2.47	2.00
350	0.42	1.52	0.36	0.30	0.29	1.35	0.40	0.29	0.45	1.50	0.40	0.29	1.28	3.38	1.21	0.94	2.34	4.64	2.35	1.90
400	0.43	1.61	0.37	0.30	0.29	1.41	0.41	0.29	0.46	1.59	0.42	0.30	1.26	3.31	1.18	0.93	2.26	4.55	2.30	1.85

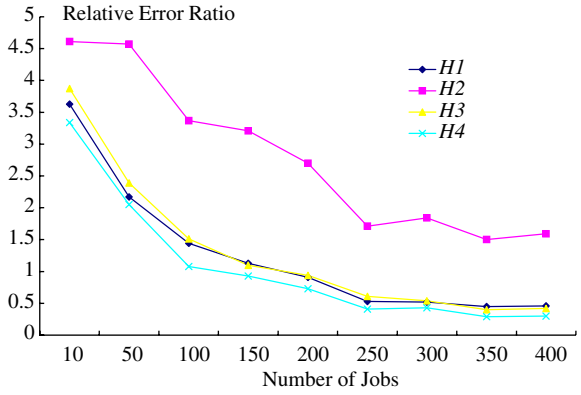

Fig. 2. Relative errors of the four heuristics with $s = 50$.

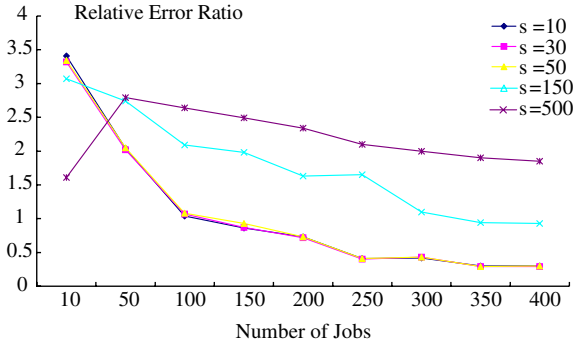

Fig. 3. Relative error ratios of *Algorithm* H_4.

[1,100]. The number of jobs n was 10, 50, 100, 150, 200, 250, 300, 350 or 400. The batch setup time s was 10, 30, 50, 150 or 500 to study the effects of different setup times on the solution quality. For each combination of n and s, all the heuristic algorithms were run over ten job sets that were generated in random. The average relative error ratios of the ten job sets for each heuristic are tabulated in Table 1. By and large, the relative error ratios are no more than 5%. From the numerical results, it is obvious that *Algorithm* H_4 outperforms all the other methods, and that *Algorithm* H_2 has a relatively inferior performance. Using the computational results of *Algorithm* H_4 as an example, we observe two significant trends. First, the relative error ratios of the heuristics decrease as the number of jobs increases (see Fig. 2). This observation suggests the practical significance of our heuristics in dealing with large-scale problems. Second, the relative error ratios of the heuristics deteriorate as the setup time becomes longer (see Fig. 3). In real-world applications, setup times are usually required to be relatively small, due to, e.g., the concept of single-digit change-over time. Therefore, our heuristic approaches should be of practical use to handle the 3MAF$/(\delta, \delta) \rightarrow \beta/C_{\max}$ problem in real-life situations.

The above results also reveal the tightness of the derived lower bounds. By transforming an instance into an ideal one that exhibits structural properties for the existence of polynomial time algorithms, we can attain a lower bound with a small deviation from the optimal solution in a reasonable time. This approach is of potential use for facilitating the development of branch-and-bound algorithms to tackle other similar combinatorial optimization problems.

6. Concluding remarks

This paper addressed the three-machine assembly-type flowshop scheduling problem with batching considerations to minimize the makespan. We first showed that the problem remains NP-hard in the strong sense even when all the jobs have the same processing time on the second-stage machine. We developed an $O(n^2)$ algorithm for optimally grouping jobs in a fixed sequence into batches. A lower bound was established through the use of a data transformation scheme and the above algorithm. To find approximate solutions to the general problem, we presented four heuristics that determine the job sequences for use in the optimal batch composition process. The computational results demonstrate the effectiveness of the heuristic algorithms and the lower bound.

For further research, it would be interesting to study the situations with more suppliers and more stages, or hybrid flowshops. To better reflect realistic supply chains, we may incorporate the transportation time/cost required to transfer a batch of parts into the model. Furthermore, centralized decision making usually calls for some sort of compensations for the parties who follow the predetermined policies. Therefore, it is worth considering rewards/compensations for coordinated production planning and scheduling as studied by Li and Xiao [24].

Acknowledgements

We are grateful to the anonymous referees for their constructive comments on earlier versions of this paper. We specially thank one of the referees who identified a derivation error in the original proof of Theorem 1.

Appendix

Proof of Theorem 1. ($\Rightarrow$) Let $A_1, A_2, \ldots,$ and A_m constitute a partition of set A. We construct a schedule of $m+1$ batches $B_1, B_2, \ldots, B_{m+1}$ as follows: the first batch B_1 contains only jobs $3m+1$, $3m+2$ and $3m+3$, and batch B_{i+1}, $1 \leqslant i \leqslant m$, contains the jobs corresponding to the elements in A_i. It is easy to verify that the makespan of the schedule is $4(m+1)\omega E$.

($\Leftarrow$) Suppose that there is an optimal schedule S for the constructed instance of $3\text{MAF}|(\delta,\delta) \to \beta,\ p_{i2} = p|C_{\max}$ with makespan no greater than $4(m+1)\omega E$. Since the total processing times of all the jobs on machine M_2 is $4(m+1)\omega E - 3(m+1)$, in schedule S, the sum of idle times and setup times on machine M_2 cannot be greater than $3(m+1)$. That is, schedule S can have at most $m+1$ batches. We first show that the first batch B_1 contains only jobs $3m+1$, $3m+2$ and $3m+3$. If the first batch B_1 contains some job i, $1 \leqslant i \leqslant 3m$, then on machine M_2 before job i there will be an idle time of at least ωE, which is greater than $3(m+1)$, a contradiction. Furthermore, if any of the jobs $3m+1$, $3m+2$ and $3m+3$ is contained in some other batch B_k with $k \neq 1$, we can transfer this job into batch B_1 without increasing the makespan. Therefore, jobs $3m+1$, $3m+2$ and $3m+3$ constitute the first batch.

As the multiplier ω is a number much larger than $3(m+1)$, any idle time caused by delayed completion on machine M_a or M_b will lead to a contradiction to the allowable idle time of $3(m+1)$. Therefore, in the following, we must ensure that no idle time is incurred. Notice that the completion times of the first batch on the three machines are 0, 0, and $3+3(4\omega E/3-1)=4\omega E$, respectively. It is clear that batch B_2 cannot contain more than three jobs, for otherwise an idle time $\sum_{i\in B_2}\omega(E+p_{ia}) - 4\omega E$, which is greater than $3(m+1)$, will be incurred. If batch B_2 is not the last batch and contains only one job, then we can transfer a job from any successor batch into batch B_2 without increasing the makespan. The same line of reasoning can be applied to any successor batch. Let k, $1 < k \leqslant m+1$, be the smallest index such that $|B_k| \geqslant 3$ and $|B_k| = 2$ for $k = 2, 3, \ldots, k-1$. For simplicity in presentation, we assume $|B_k| = 3$. The other case where B_k has more than three jobs can be similarly analyzed. We examine the following three completion times to show that index k must be 2.

1. Completion time of batch B_k on M_a:

$$\omega\left(\left(2(k-2)E + \sum_{i\in B_2\cup\cdots B_{k-1}} p_{ia}\right) + \left(3E + \sum_{i\in B_k} p_{ia}\right)\right) = \omega\left((2k-1)E + \sum_{i\in B_2\cup\cdots B_k} p_{ia}\right). \quad (1)$$

2. Completion time of batch B_k on M_b:

$$\omega\left(\left(4(k-2)E - 2\sum_{i\in B_2\cup\cdots B_{k-1}} p_{ia}\right) + \left(6E - 2\sum_{i\in B_k} p_{ia}\right)\right) = \omega\left((4k-2)E - 2\sum_{i\in B_2\cup\cdots B_k} p_{ia}\right). \quad (2)$$

3. Completion time of batch B_{k-1} on M_2:

$$4\omega E + (k-2)(4\omega E/3 - 1) = 4(k+1)\omega E/3 - (k-2). \quad (3)$$

To avoid incurring idle time before batch B_k on machine M_2:

$$4(k+1)\omega E/3 - (k-2) - \omega\left((2k-1)E + \sum_{i\in B_2\cup\cdots B_k} p_{ia}\right) \geqslant 0$$

and

$$4(k+1)\omega E/3 - (k-2) - \omega\left((4k-2)E - 2\sum_{i\in B_2\cup\cdots B_k} p_{ia}\right) \geqslant 0$$

must be simultaneously satisfied. The inequality

$$4(k+1)\omega E/3 - (k-2) - \omega\left((2k-1)E + \sum_{i\in B_2\cup\cdots B_k} p_{ia}\right) \geqslant 0$$

can be simplified to

$$\omega\left(7E/3 - 2kE/3 - \sum_{i\in B_2\cup\cdots B_k} p_{ia}\right) - (k-2) \geqslant 0.$$

Because ω is much larger than $k-2$, we have

$$7E/3 - 2kE/3 - \sum_{i\in B_2\cup\cdots B_k} p_{ia} > 0,$$

or

$$7E/3 - 2kE/3 - \sum_{i\in B_2\cup\cdots B_k} p_{ia} = 0 \quad \text{and} \quad k \leqslant 2.$$

If it is the latter case, then we have shown that $k=2$. Therefore, we assume that

$$7E/3 - 2kE/3 - \sum_{i\in B_2\cup\cdots B_k} p_{ia} > 0. \quad (4)$$

Similarly, the inequality

$$4(k+1)\omega E/3 - (k-2) - \omega\left((4k-2)E - 2\sum_{i\in B_2\cup\cdots B_k} p_{ia}\right) \geqslant 0$$

implies

$$10E/3 - 8kE/3 + 2\sum_{i\in B_2\cup\cdots B_k} p_{ia} > 0,$$

or

$$10E/3 - 8kE/3 + 2\sum_{i\in B_2\cup\cdots B_k} p_{ia} = 0 \quad \text{and} \quad k \leqslant 2.$$

We similarly assume that

$$10E/3 - 8kE/3 + 2\sum_{i\in B_2\cup\cdots B_k} p_{ia} > 0. \quad (5)$$

Combining (4) and (5), we have $10E/3 - 8kE/3 + 14E/3 - 4kE/3 > 0$, which leads to $k < 2$. But the premise of the above derivation is that $k > 1$, a contradiction. So, we have obtained the fact that the second batch B_2 must contain exactly three jobs. Next, we examine the constituent jobs in batch B_2. If $\sum_{i\in B_2} p_{ia} = (3E + \sum_{i\in B_2} x_i)\omega > 4\omega E$, then the completion time on M_a will lead to a non-zero idle time on machine M_2. On the other hand, if $\sum_{i\in B_2} p_{ib} = (6E - 2\sum_{i\in B_2} x_i) > 4\omega E$, then the completion time on M_b will also lead to a non-zero idle time on machine M_2. Therefore, to avoid idle time, the equality $\sum_{i\in B_2} p_{ia} = 4\omega E$ must hold, i.e., the elements corresponding to the three jobs of batch B_2 must sum to exactly E. Let the three elements form subset A_1. The completion times of batch B_2 on the three machines are $4\omega E$, $4\omega E$ and $8\omega E$, respectively. Therefore, we can apply the same analysis process and successively construct subsets $A_2, A_3, \ldots,$ and A_m. □

References

[1] Johnson SM. Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1954;1:61–7.

[2] Pinedo M. Scheduling theory, algorithms, and systems. Englewood Cliffs, NJ: Prentice Hall; 2001.

[3] Reisman A, Kumar A, Motwani J. Flowshop scheduling/sequencing research: a statistical review of the literature, 1952–1994. IEEE Transactions on Engineering Management 1997;44:316–29.

[4] Lee CY, Cheng TCE, Lin BMT. Minimizing makespan in the 3-machine assembly-type flowshop scheduling problem. Management Science 1993;39:616–25.

[5] Garey MR, Johnson DS, Sethi R. The complexity of flowshop and job shop scheduling. Mathematics of Operations Research 1976;1:117–29.

[6] Bhatnagar R, Chandra P, Goyal SK. Models for multi-plant coordination. European Journal of Operational Research 1993;67:141–60.

[7] Sun X, Morizawa K, Nagasawa H. Powerful heuristics to minimize makespan in fixed, 3-machine, assembly-type flowshop scheduling. European Journal of Operational Research 2003;146:498–516.

[8] Potts CN, van Wassenhove LN. Integrating scheduling with batching and lot-sizing a review of algorithms and complexity. Journal of the Operational Research Society 1992;43:95–406.

[9] Webster S, Baker KR. Scheduling groups of jobs on a single machine. Operations Research 1995;43:692–703.

[10] Allahverdi T, Gupta JND, Aldowaisan T. A review of scheduling research involving setup considerations. Omega 1999;27:219–39.
[11] Cheng TCE, Gupta JND, Wang G. A review of flowshop scheduling research with setup times. Production and Operations Management 2000;9:262–82.
[12] Potts CN, Kovalyov MK. Scheduling with batching: a review. European Journal of Operational Research 2000;102:228–49.
[13] Lee CY, Uzsoy R, Martin-Vega LA. Efficient algorithms for scheduling semiconductor burn-in operations. Operations Research 1992;40:764–75.
[14] Ahmadi JH, Ahmadi RH, Dasu S, Tang CS. Batching and scheduling jobs on batch and discrete processors. Operations Research 1992;39:750–63.
[15] Albers S, Brucker P. The complexity of one-machine batching problems. Discrete Applied Mathematics 1993;47:87–107.
[16] Coffman Jr EG, Yannakakis M, Magazine MJ, Santos C. Batch sizing and job sequencing on a single machine. Annals of Operations Research 1990;26:135–47.
[17] Santos CA, Magazine MJ. Batching in single operation manufacturing systems. Operations Research Letters 1985;4: 99–103.
[18] Cheng TCE, Wang Q. Batching and scheduling to minimize the makespan in the two-machine flowshop. IIE Transactions 1998;30:447–53.
[19] Cheng TCE, Lin BMT, Toker A. Makespan minimization in the two-machine flowshop batch scheduling problem. Naval Research Logistics 2001;47:128–44.
[20] Glass CA, Potts CN, Strusevich VA. Scheduling batches with sequential job processing for two-machine flow and open shops. INFORMS Journal on Computing 2001;13:120–37.
[21] Kovalyov MY, Potts CN, Strusevich VA. Batching decisions for assembly production systems. European Journal of Operational Research 2004;157:620–42.
[22] Lawler EL, Lenstra JK, Rinnooy Kan AHG, Smoys DB. Sequencing and scheduling: algorithms and complexity. In: Graves SC, Rinnoy Kan AHG, Zipkin P, editors. Handbooks of operations research and management science, Volume 4: logistics of production and inventory. Amsterdam: North Holland; 1993. p. 445–522.
[23] Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. New York: Freeman; 1979.
[24] Li C-L, Xiao W-Q. Lot streaming with supplier-manufacturer coordination. Naval Research Logistics 2004;51:467–631.