Int. J. Production Economics 139 (2012) 473–483

Contents lists available at SciVerse ScienceDirect

Int. J. Production Economics

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

A case study of batch scheduling for an assembly shop

Rock Lin, Ching-Jong Liao*

Department of Industrial Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan

A R T I C L E   I N F O

Article history:
Received 11 February 2011
Accepted 13 April 2012
Available online 1 June 2012

Keywords:
Mixed integer programming
Job dividing
Batch processing
Family scheduling
Lower bound

A B S T R A C T

This paper addresses a scheduling problem for a two-stage assembly shop in a machinery factory. At stage one, all parts of jobs are assembled simultaneously on a batch machine with a common processing time and a constant batch setup time. Then the assembled jobs are moved to the second stage to perform system integration with different processing times on a discrete machine. On both machines, a family setup time is required when the processing switches from one family to a different one. The objective is to minimize the weighted sum of makespan, total completion time and total tardiness. A Mixed Integer Programming (MIP) model is developed for solving small-size problems, and three heuristics are proposed for solving medium- and large- size problems. Computational experiments show that RFBFS, a full batch family sorting heuristic combining with rolling horizon scheduling strategy, is better than the other two heuristics in terms of solution quality. Real-life implementation also shows that the performance of RFBFS is significantly better than the current method.

1. Introduction

In this paper, we study a batch scheduling problem for a two-stage assembly shop which is encountered frequently in machinery factories. The problem is similar to the two-stage assembly scheduling problem in the literature (Yokoyama, 2001; Lin and Cheng, 2002; Koulamas and Kyparisis, 2007; Sung and Kim, 2008), where raw materials are fabricated into parts in the first stage and then all the parts are assembled into final products in the second stage. However, the fabrication of parts becomes more difficult and expensive due to the complicated technology and processes, and hence outsourcing of parts (purchasing) has been a routine procedure. Therefore, we model the assembly shop as a two-stage flow shop which consists of module assembly in the first stage and system integration in the second stage.

In the past two decades, technology has kept developing and there has been a need to increase diversity. The combination of these two trends results in multi- function and high-performance products, which makes the production of machinery much more complicated than before. In such an environment, the machinery factories face the following scheduling problems in their shop management:

(1) How to reduce cycle time?
(2) How to reduce work-in-process (WIP)?
(3) How to fulfill timely delivery?
(4) How to deal with insertion of rush orders?

This paper is motivated from a machinery factory in Taiwan, Hemingstone Machinery Co., Ltd., which produces over 10 different products of plastic bag making machines for customer orders. Currently, the orders are sequenced by the EDD rule. To respond quickly to the customer needs, a product (job) is assembled simultaneously by several workers within a frame (see Fig. 1). Due to the space limitation, idle times and waiting times easily occur resulting from the interference of workers. In order to reduce the idle times and waiting times, we propose the strategies of job dividing (derived from the concept of lot splitting) and batch processing to make the shop more efficient. To shorten the completion time, we also perform a strategy of family scheduling where jobs in the same family are grouped into batches to eliminate the family setups (Liao and Liao, 2008; Schaller and Gupta, 2008). As depicted in Fig. 2, the first stage is modeled as a batch machine so that c skilled workers can assemble c jobs separately and simultaneously. The second stage is modeled as a discrete machine in which jobs entering in batches are processed sequentially and leave one by one. As such, most of the idle times due to the interference with each other in the first stage can be diminished, and family scheduling can reduce the job setups and family setups in both stages.

Objective functions are used to evaluate the performance of a scheduling solution approach. In this paper, jobs in a batch share a batch setup and jobs within a family share a family setup. Such batching of jobs may delay the processing of other jobs and cause them to be tardy. To resolve this trade-off, we use a

* Correspondence to: Department of Industrial Management, National Taiwan University of Science and Technology, 43 Keelung Road, Section 4, Taipei 106, Taiwan. Tel.: +886 2 27376337.
E-mail address: cjliao@mail.ntust.edu.tw (C.-J. Liao).

0925-5273/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ijpe.2012.05.002

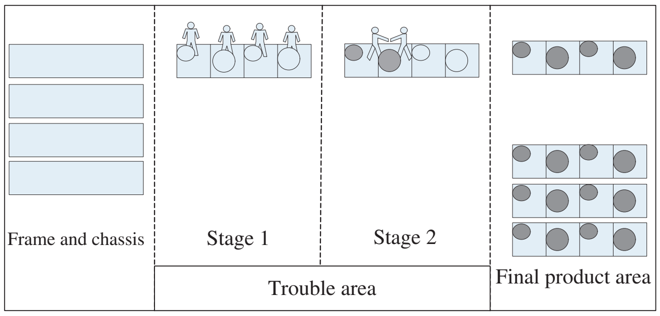

Fig. 1. Work-flow of the current manufacturing process.

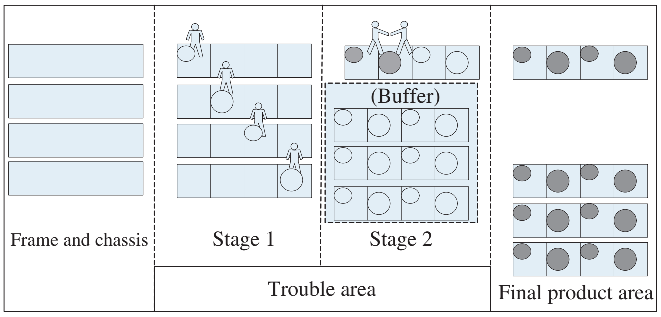

Fig. 2. Work-flow with job dividing and batch processing.

multi-objective of makespan, total completion time and total tardiness, which consist of the conflicting measures to evaluate the performance of the proposed solution approach (Framinan et al., 2002; Amorim et al., 2012).

Batch processing is one of the major characteristics of the considered problem. Potts and Kovalyov (2000) provide a review for batch scheduling which reflects the extensive literature on models that integrate scheduling with batching decisions. Lee et al. (1992), Webster and Baker (1995), Potts et al. (2001) and Neale and Duenyas (2000) model a burn-in oven as a batch processing machine and develop methods to obtain better performance in terms of cycle time and/or flow time. A batch processing machine can process several jobs simultaneously and the processing length of a batch is equal to the largest processing time among all the jobs in a batch. Ahmadi et al. (1992) first consider a manufacturing system equipped with batch and discrete machines, where each batch machine can process up to c jobs simultaneously. They propose the Full Batch-SPT and Full Batch-Dealing policies for solving the problems of minimizing the total completion time. They also propose the LPT-Johnson rule for minimizing makespan on the system that processes multiple families of jobs. Hoogeveen and van de Velde (1998) extend the work of Ahmadi et al. present a useful Lagrangian lower bound by using the concept of positional completion times. Kim and Kim (2002) also address the batch-discrete system scheduling problem to minimize the total completion time. Gong et al. (2010) consider a two-stage flow shop scheduling problem where the first machine is a batching machine subject to the blocking constraint and the second machine is a discrete machine with shared setup times. They show that the problem is NP-hard when the objective is to minimize the makespan. Liu et al. (2009) study the problem of scheduling jobs with equal processing times on a single batch machine. They develop some useful algorithms for minimizing the total weighted completion time, maximum lateness and total tardiness. Besides, Schaller (2007) shows a single machine problem to minimize total tardiness when family setup times exist to be NP-hard since the single machine case to minimize total tardiness is already NP hard, proved by Du and Leung (1990).

In summary, the considered scheduling problem is modeled as a two-machine flow shop system, consisting of a batch machine with common processing time and constant batch setup time followed by a discrete machine with different processing times. On both machines a family setup time is needed when the processing switches from one family to a different one. The scheduling objective is to minimize the sum of weighted makespan, total completion time and total tardiness. This problem is NP-hard since the special case of the problem to minimize total completion time without family setups has been proven to be NP-hard by Ahmadi et al. (1992). The NP hardness of the problem can also be derived from the fact that a single machine problem to minimize total tardiness, which is a special case of the problem, is NP-hard proved by Du and Leung (1990).

The rest of the paper is organized as follows. In Section 2, we describe the current scheduling method in the case factory. In Section 3, we integrate the concept of job dividing into the batch scheduling model and present a Mixed Integer Programming (MIP) model for this problem. We also develop three heuristics for medium- and large-size problems. In Section 4, computational experiments and real-life implementation are provided to evaluate the proposed heuristics. Finally, conclusions are provided in Section 5.

2. Current scheduling approach

In this section, we introduce the notation and describe the current scheduling approach employed in the case factory.

2.1. Notation and assumptions

To describe the problem, we introduce the following notation:

j	job index
i	batch index
r	position index in the batch
b	number of batches in stage 1, $b \geq 2$
c	batch size in stage 1
J	set of jobs to be processed, $J=\{1,2,\ldots,n\}$
$p_{a\ j}$	processing time of J_j for one-piece processing in stage 1, $p_{a\ j}=p_a$
p_{1j}	processing time of J_j in stage 1, $p_{1j}=p_1$
p_{2j}	processing time of J_j in stage 2
s_a	job setup time, preparing for one-piece processing in stage 1
s_b	batch setup time, preparing for batch processing in stage 2
s_{f1}	family setup time in stage 1
s_{f2}	family setup time in stage 2
f_j	family of J_j, $f_j=1,2,3,4$
$f_{1,[i,r]}$	family of job $[i,r]$, the job scheduled in the rth position of batch i,in stage 1
$f_{2,[j]}$	family of $J_{[j]}$, the job scheduled in the jth position, in stage 2
$k_{1,[i,r]}$	$= \begin{cases} 0 \text{ if job } [i,r] \text{ and its predecessor in the first stage belong to the same family} \\ 1 \text{ otherwise} \end{cases}$
$k_{2,[j]}$	$= \begin{cases} 0 \text{ if } J_{[j]} \text{ and its predecessor in the second stage belong to the same family} \\ 1 \text{ otherwise} \end{cases}$
S	set of scheduled jobs
S_U	set of unscheduled jobs

$\mathcal{B}$	set of scheduled batches
B_i *i*th	batch in stage 1, $B_i \subset \mathcal{B}$
$p_1(B_i)$	processing time of B_i in stage 1
D_j	delivery date of J_j
d_j	due date of J_j
$d_{[j]}$	due date of $J_{[j]}$ in the stage 2
$C_{1,[i]}$	completion time of batch *i* in stage 1
$C_{2,[j]}$	completion time of $J_{[j]}$ in stage 2
T_j	tardiness of $J_{[j]}$

The following assumptions are made for the problem:

- All jobs are available at time zero.
- The total number of jobs is divisible by the batch size *c*, i.e., $n = b \times c$.
- A job in stage 2 can start processing only after its batch is finished in stage 1.
- A job setup time s_a is required for one-piece processing in stage 1.
- A batch setup time s_b is required for batch processing in stage 1.
- A family setup time s_{f1} is required before starting the processing of a new family in stage 1.
- A family setup time s_{f2} is required before starting the processing of a new family in stage 2.
- The processing times, setup times and delivery dates are known and fixed.

2.2. Manufacturing process

In order to simplify the process and enlarge the capacity of production, all parts and sub-assemblies are outsourced to qualified partners or vendors. Therefore, the production system can be modeled as an assembly flow shop, which consists of two stages: (1) an assembly stage which assembles parts and modules into the frame and (2) an integration stage which implements the integration of mechanic and electronic engineering (M&E). To allow all products to share a common assembly line and to have a common processing time, modular design and standardization are performed so that all products are composed of eleven sub-systems as given in Table 1. To balance the line and shorten the completion time, the load and the positions of the operations for the sub-systems are considered (Nearchou, 2011). All operations are partitioned into four operation blocks (i.e., A, B, C, D) and are assigned respectively to a team of four skilled workers. To start processing a job, the four workers move the frame and chassis together to the area in stage 1, and they prepare the required parts and tools for the job. Then they perform assembling in each block. Due to the limited space, the assembling is easily interfered with each other, which leads to substantial waiting time and often results in a bottleneck. Once the assembly is finished in stage 1, the job is transferred to stage 2, in which there is a team of two skilled workers. They implement the mechanical and electronic integration which includes servo or computer system installing as well as parameters setting and testing. After that, the finished job is moved to the final product area.

2.3. The setup times and the families

The majority of scheduling research considers setup times as negligible or part of processing time (Logendran et al., 2005; Allahverdi et al., 1999), but in this case they are significant and have to be considered. The setup times here can be classified into two types: (1) job setup which is used to prepare and position all of the needed parts for the job to be processed, and (2) family setup which is used to obtain tools, set jigs and fixtures and check parts for processing jobs in different families (Eren, 2007). Before the operations are started in stage 1, a job setup time s_a is needed while family setups s_{f1} and s_{f2} are incurred on both stages when starting the first job or switching to a job in a different family. For simplicity, we assume that they are constant for all jobs with $s_a = 5.2$ h, $s_{f1} = 3.2$ h and $s_{f2} = 1.6$ h. As listed in Table 2, there are currently over 10 models produced in the case factory, and the jobs can be grouped into four families ($f_1 \sim f_4$) based on model attribute. Note that the job setup time s_a is not included in the job processing time since it will be compared with the batch setup time s_b which is required for the batch processing.

2.4. The objective function

As discussed in the introduction, the problem is better represented by multiple objectives with the weighted sum of makespan, total completion time and total tardiness, i.e.,

$$Z = \alpha C_{\max} + \beta \sum_{j=1}^{n} C_{2,[j]} + \gamma \sum_{j=1}^{n} T_j$$

where α, β and γ are the nonnegative weight parameters, which can be determined in the following manner:

(1) $C_{\max}$ and $\sum C_{2,[j]}$ do not have the same variation range in nature. To balance this difference, we adopt the adjustment procedure presented by Framinan et al. (2002). That is, we calculate the expected completion time of a job as $C_{\max}/2$, assuming *n* identical jobs, and hence $nC_{\max}/2 \approx \sum C_{2,[j]}$. Thus, we multiply $C_{\max}$ and $\sum C_{2,[j]}$ by $n/2$ and 1, respectively.
(2) In practice, the determination of weight parameters is strongly dependent on the preferences of the decision maker

Table 2
The models and families of jobs produced in the last two years.

Model		Quantity		Processing time (h)	
No.	Family	2007	2008	p_1/1 worker	p_2/2 workers
900BR	f_1	26	15	24	16
1000DT	f_1	30	46	24	16
800ST2	f_2	44	51	24	12
1100ST3	f_2	18	11	24	12
1000TT	f_3	43	30	24	8
800TT	f_3	37	31	24	8
1000VV	f_4	21	47	24	10
Others	f_4	49	67	24	10
Average (month)		22.33	24.83	–	–

Table 1
The eleven sub-systems and the four blocks (A, B, C and D).

Front section		Middle section		Rear section	
A-1	Pushing roller unit	**B**-1	Horizontal EPC unit	**D**-1	Bag paddle unit
A-2	Tension roller unit	**B**-2	Vertical EPC unit	**D**-2	Punching unit
A-3	Pressing bag unit	**B**-3	Sealing knife unit	**D**-3	Conveyor unit
A-4	Sending material unit	**C**-1	Rocking system		

(Taboada and Coit, 2008). In the case factory, the total completion time helps to measure on-time delivery, while the total tardiness mainly serves as the purpose for overtime planning. The management considers the total completion time and the total tardiness are double of the makespan. So, $C_{\max}$, $\sum C_{2,[j]}$ and $\sum T_j$ are further multiplied by 1, 2 and 2, respectively. For $n=12$, the weights are $(\alpha,\beta,\gamma)=(6,2,2)=(0.6,0.2,0.2)$.

2.5. Current scheduling approach

The work-flow of the current scheduling approach is given in Fig. 3. After the production manager confirms the delivery date, the salesman accepts the customer order and sends the order notice to the production manager. Once the order notice arrives at the shop floor, the MRP center releases the purchase order (PO) and manufacturing order (MO), in which the due dates (d_j) are usually defined as a week, 5 days $\times$ 8 h$=$40 h, before the delivery days (D_j), i.e., $d_j=D_j-40$. All purchase orders are immediately placed to suppliers on the basis of PO, meanwhile the schedule is made on the basis of MO. Currently, the sequence of jobs is determined by the EDD dispatching rule, and one-piece flow (Li and Rong, 2009) is performed in both stages with the same job sequence.

When setup times are zero, the EDD rule is optimal for minimizing the maximum lateness (Baker and Magazine, 2000). However, many job setups and family setups are incurred for one-piece processing in the current shop, so the EDD rule usually results in a large completion time. Moreover, with one-piece flow the four workers operate simultaneously in a limited space (see Fig. 1), which interferes with each other, and the resulting significant idle times make the processing time become $p_a=9$ h rather than 24 h/4$=$6 h (see Table 2). Therefore, the following problems emerge from the current scheduling approach:

(1) One-piece flow easily results in material shortage and starvation, which render the processing difficult to meet the schedule.
(2) Frequent setups and idles cause longer flow time and more overtime.
(3) It is unable to deal with rush orders due to the low on-time completion rate.

To deal with these problems, we propose the strategies of job dividing and batch processing, which will be discussed in the following section.

3. Development of solutions methods

In this section, we develop the solution methods for the considered problem. First, we introduce the concept of job dividing and combine it with batch processing so as to reduce the setup times and improve the efficiency of the assembly shop. Then we develop two optimal properties and present a Mixed Integer Programming (MIP) formulation for deriving an optimal solution. Finally, we propose three heuristic methods for obtaining a near optimal solution.

3.1. Job dividing and batch processing

To improve the current scheduling method, we extend the concept of lot splitting (Bukchin et al., 2002) and propose the job dividing strategy to divide a job into several sub-jobs. Combining the job dividing strategy with batch processing, the sub-jobs of c jobs are processed separately and simultaneously by c workers. This can avoid interference and diminish the waiting time. Since the jobs in a batch enter and leave the machine continuously, the c workers must stop processing at the same time; otherwise some workers will be idle while others are still busy. In order to ensure that the c workers stop processing simultaneously, modular design and standardization are performed such that all jobs have a common processing time and near equal workloads are assigned to c workers. If necessary, the four workers can support each other so that the sub-jobs can be finished at the same time. As such, the processing will become smooth and no bottleneck will exist in the first stage. As depicted in Fig. 2, the system can then be considered as a two-stage flow shop where the first stage has a batch machine with four workers and the second stage has a discrete machine with two workers.

Following the previous description, in the first stage we divide each job into c sub-jobs that must be processed simultaneously by c workers which can be regarded as a batch machine with capacity c jobs (Liu and Yu, 2000). The processing length of batch i is the largest processing time of the jobs in the batch, i.e., $p_1(B_i)=\max p_{1j}=p_1$. In addition, a batch setup s_b is needed whenever a batch is formed, while a family setup s_{f1} is required for starting the first job of the first batch or switching to a job in a different family. Note that a family setup is not required when the last job of a batch and the first job of the next batch are within the same family. Then the completion time of batch i, $C_{1,[i]}$, is equal to the starting time of the batch plus the batch setup time and the family setup times in the batch as well as the processing time of this batch, i.e.,

$$C_{1,[0]}=0 \tag{1}$$

$$C_{1,[i]}=C_{1,[i-1]}+s_b+s_{f1}\sum_{r=1}^{c}k_{1,[i,r]}+p_1, \quad i=1,\ldots,b$$
$$(k_{1,[i,r]}=0 \text{ if } f_{1,[i,r]}=f_{1,[i,r-1]}, \text{ otherwise } k_{1,[i,r]}=1) \tag{2}$$

In the second stage the jobs are processed by a discrete machine, where the jobs enter in batch but leave one by one. A family setup s_{f2} is required for starting the first job or switching

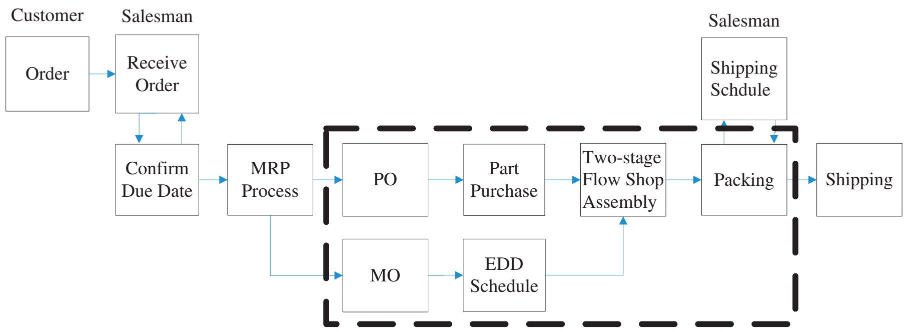

Fig. 3. The work-flow of the current scheduling approach.

to a job in a different family. The completion time of $J_{[j]}$, $C_{2,[j]}$, is equal to $C_{1,[i]}$ or $C_{2,[j-1]}$ plus the family setup time and the processing time of $J_{[j]}$. That is,

$$C_{2,[0]}=0 \tag{3}$$

$$C_{2,[j]}=\max C_{1,[i]},C_{2,[j-1]}+s_{f2}\times k_{2,[j]}+p_{2,[j]},\quad i=1,\ldots,b,j=1,\ldots,n \\ (k_{2,[j]}=0 \text{ if } f_{2,[j]}=f_{2,[j-1]}, \text{ otherwise } k_{2,[j]}=1) \tag{4}$$

Then,

$$C_{\max}=C_{2,[n]} \tag{5}$$

$$\sum_{j=1}^{n}T_j=\sum_{j=1}^{n}\max\{C_{2,[j]}-d_{[j]},0\} \tag{6}$$

$$Z=\alpha C_{\max}+\beta\sum_{j=1}^{n}C_{2,[j]}+\gamma\sum_{j=1}^{n}T_j. \tag{7}$$

To improve the on-time completion rate, we accept only the schedule whose total tardiness is less than 20 h a week since the tardiness can be resolved by the limited overtime during the weekend. If not, the production manager negotiates with the salesman to change the due dates of the tardy jobs and re-schedules the jobs.

3.2. Property development

Define a batch containing exactly c jobs as a full batch, and a non-full batch as a partial batch. Recall that we assume the number of jobs is a multiple of the batch size (i.e., $n=b\times c$).

Property 1. For the considered problem, all batches must be full in an optimal schedule.

Proof. Let S' be a sequence that contains some full batches and some partial batches, i.e., $\mathcal{B}_1,\mathcal{B}_2,\ldots,\mathcal{B}'_k,\mathcal{B}'_{k+1},\ldots,\mathcal{B}'_{b'}$. We move jobs from $\mathcal{B}'_{k+1}$, the second partial batch, to make $\mathcal{B}'_k$, the first partial batch, to be a full batch $\mathcal{B}_k$. Repeat this process until we obtain a schedule S that contains all full batches ($\mathcal{B}_1,\mathcal{B}_2,\ldots,\mathcal{B}_k,\mathcal{B}_{k+1},\ldots,\mathcal{B}_b$). Then the starting times of the jobs in S is either smaller than in S' (for those jobs after $\mathcal{B}'_k$) or the same as in S' (for the rest of jobs), and so are the completion times. Thus, S' is dominated by S in terms of the objective function value, which completes the proof.

Property 2. In an optimal schedule, jobs from the same family in a batch must be processed consecutively.

Proof. The reader can refer to Baker (1999) and Chandru et al. (1993).

With these two properties, we can limit our search, in the MIP model and the heuristics, to find an optimal or near optimal schedule with the number of jobs being a multiple of batch size. This results in a significant saving in computation time. Furthermore, we adopt an 80/20 of mixed MTO/MTS policy to deal with the insertion of rush orders. The production manager can thus select the jobs from MTS to form a full batch when the jobs from MTO are not enough to form a full batch.

3.3. MIP formulation

The variables used in the MIP formulation are defined as follows:

- *Job-batch assignment*
  $X_{j,[i,r]}$ binary variables taking value 1 if J_j is scheduled in the rth position of batch $i$in stage 1, and 0 otherwise.
- *Family setup*
  $k_{1,[i,r]}$ binary variable taking value 1 if a family setup is required for the job scheduled in the rth position of batch i in stage 1, and 0 otherwise.
  $k_{2,[j]}$ binary variable taking value 1 if a family setup is required for the job scheduled in the jth position in stage 2 and 0 otherwise.
- *Auxiliary variables*
  $g_{i,[r,l]}$ binary variable used to derive $k_{1,[i,r]}$.

Also, completion time and tardiness variables are non-negative continuous variables. Then, based on Eqs. (1)–(7) and the two properties, the problem can be formulated as the following MIP model

$$\text{Minimize}\ \alpha C_{\max}+\beta\sum_{j=1}^{n}C_{2,[j]}+\gamma\sum_{j=1}^{n}T_j \tag{8}$$

subject to

$$\sum_{j=1}^{n}\sum_{r=1}^{c}X_{j,[i,r]}=c,\quad i=1,\ldots,b \tag{9}$$

$$\sum_{i=1}^{b}\sum_{r=1}^{c}X_{j,[i,r]}=1,\quad j=1,\ldots,n \tag{10}$$

$$\sum_{j=1}^{n}X_{j,[i,r]}=1,\quad i=1,\ldots,b,\ r=1,\ldots,c \tag{11}$$

$$f_{1,[i,r]}=\sum_{j=1}^{n}X_{j,[i,r]}f_j,\quad i=1,\ldots,b,\ r=1,\ldots,c \tag{12}$$

$$f_{2,[(i-1)\times c+r]}=\sum_{j=1}^{n}X_{j,[i,r]}f_j,\quad i=1,\ldots,b,\ r=1,\ldots,c \tag{13}$$

$$d_{[(i-1)\times c+r]}=\sum_{j=1}^{n}X_{j,[i,r]}d_j,\quad i=1,\ldots,b,\ r=1,\ldots,c \tag{14}$$

$$p_{2,[(i-1)\times c+r]}=\sum_{j=1}^{n}X_{j,[i,r]}p_{2j},\quad i=1,\ldots,b,\ r=1,\ldots,c \tag{15}$$

$$k_{1,\,1,1}=1 \tag{16}$$

$$f_{1,[i,r]}-f_{1,[i,r-1]}=-3g_{i,[r,1]}-2g_{i,[r,2]}-g_{i,[r,3]}+g_{i,[r,5]} \\ +2g_{i,[r,6]}+3g_{i,[r,7]},\quad i=1,\ldots,b,\ r=2,\ldots,c \tag{17}$$

$$g_{i,[r,1]}+g_{i,[r,2]}+g_{i,[r,3]}+g_{i,[r,4]}+g_{i,[r,5]}+g_{i,[r,6]}+g_{i,[r,7]}=1, \\ i=1,\ldots,b,\ r=2,\ldots,c \tag{18}$$

$$k_{1,[i,r]}=g_{i,[r,1]}+g_{i,[r,2]}+g_{i,[r,3]}+g_{i,[r,5]} \\ +g_{i,[r,6]}+g_{i,[r,7]},\quad i=1,\ldots,b,\ r=2,\ldots,c \tag{19}$$

$$f_{1,[i,1]}-f_{1,[i-1,c]}=-3g_{i,[1,1]}-2g_{i,[1,2]}-g_{i,[1,3]}+g_{i,[1,5]} \\ +2g_{i,[1,6]}+3g_{i,[1,7]},\quad i=2,\ldots,b \tag{20}$$

$$g_{i,[1,1]}+g_{i,[1,2]}+g_{i,[1,3]}+g_{i,[1,4]}+g_{i,[1,5]}+g_{i,[1,6]}+g_{i,[1,7]}=1,\quad i=2,\ldots,b \tag{21}$$

$$k_{1,[i,1]}=g_{i,[1,1]}+g_{i,[1,2]}+g_{i,[1,3]}+g_{i,[1,5]}+g_{i,[1,6]}+g_{i,[1,7]},\quad i=2,\ldots,b \tag{22}$$

$$k_{2,[(i-1)c+r]}=k_{1,[i,r]},\quad i=1,\ldots,b,\ r=1,\ldots,c \tag{23}$$

$$C_{1,[0]}=0 \tag{24}$$

$$C_{1,[i]} = C_{1,[i-1]} + s_b + s_{f1}\sum_{r=1}^{c} k_{1,[i,r]} + p_1, \quad i=1,...,b \tag{25}$$

$$C_{2,[0]} = 0 \tag{26}$$

$$C_{2,[(i-1)c+r]} = \max\{C_{1,[i]}, C_{2,[(i-1)c+r-1]}\} + s_{f2}k_{2,[(i-1)c+r]} + p_{2,[(i-1)c+r]}, \quad i=1,...,b-1,\ r=1,...,c \tag{27}$$

$$C_{\max} = C_{2,[n]} \tag{28}$$

$$T_j = \max C_{2,[j]} - d_{[j]}, 0, \quad j=1,2,...,n \tag{29}$$

$$X_{j,[i,r]}, k_{1,[i,r]}, k_{2,[j]} = 0 \text{ or } 1,\ j=1,2,...,n\ \ i=1,...,b,\ r=2,...,c \tag{30}$$

$$g_{\ i,[r,l]} = 0 \text{ or } 1, \quad i=1,...,b,\ r=2,...,c,\ l=1,...,7 \tag{31}$$

Constraints (27) and (29) are not linear, but they can be easily transformed into linear form. For example, constraint (27) can be rewritten as

$$C_{2,[(i-1)c+r]} \geq C_{1,[i]} + s_{f2}k_{2,[(i-1)c+r]} + p_{2,[(i-1)c+r]}$$

$$C_{2,[(i-1)c+r]} \geq C_{2,[(i-1)c+r-1]} + s_{f2}k_{2,[(i-1)c+r]} + p_{2,[(i-1)c+r]}$$

We now elaborate on the MIP formulation. Objective function (8) is to minimize the weighted sum of makespan, total completion time, and total tardiness. Constraint (9) ensures that each batch consists of exactly c jobs. Constraints (10) and (11) ensure that each job is scheduled in exactly one position of a batch and each position is scheduled exactly one job, respectively. Constraint (12) determines the job family scheduled in the rth position of batch $i$in stage 1. Constraints (13), (14) and (15) determine the job family, due date and processing time scheduled in the$j$th position in stage 2. Constraints (16)–(22) determine the value of family setup variable $k_{1,[i,r]}$. As the value on the left-hand side of constraint (17) are not 0–1, i.e., $f_{1,[i,r]} - f_{1,[i,r-1]} = \{-3,-2,-1,0,1,2,3\}$, the auxiliary variables $g_{i,[r,l]}$ and constraints (17)–(22) are added to the model to make it as an MIP problem (Hillier and Lieberman, 2001). If job $[i,r]$ and its predecessor in stage 1 belong to the same family, then $k_{1,[i,r]}=0$, otherwise $k_{1,[i,r]}=1$. Constraint (23) determines the values of family setup variables $k_{2,[j]}$. Since all jobs flow through the system in the same order, so $k_{2,[j]}=k_{1,[i,r]}$. Constraints (24) and (25) define the completion times of batches in stage 1. Constraints (26) and (27) define the completion times of jobs in stage 2, and ensure that a job can start only after the job and its predecessor have finished processing in stage 1. Constraints (28) and (29) define the performance measures of the problem. Constraints (30) and (31) are binary variable restrictions.

Example 1. As an illustration, consider a 12-job problem with f_j=(3,2,3,4,1,4,2,2,3,3,4,4), d_j=(100,160,160,100,180,140,140,140, 140,140,180,140), s_b =6.4, s_{f1} =3.2, s_{f2} =1.6. Other data are given in Table 2.

Solving the above MIP model by LINGO 10 software, we obtain the optimal solution as follows:

The objective value with (α,β,γ)=(0.6,0.2,0.2)

$$Z = 331.04,\ C_{\max} = 164.0,\ \sum_j C_j = 1163.2,\ T_{\max} = 0$$

The assignment variables

$X_{10,[1,1]}=1,\ X_{3,[1,2]}=1,\ X_{1,[1,3]}=1,\ X_{9,[1,4]}=1;$
$X_{4,[2,1]}=1,\ X_{11,[2,2]}=1,\ X_{12,[2,3]}=1,\ X_{6,[2,4]}=1;$
$X_{8,[3,1]}=1,\ X_{7,[3,2]}=1,\ X_{2,[3,3]}=1,\ X_{5,[3,4]}=1.$

All other assignment variables are zero.

The family setup variables

$k_{1,[1,1]}=1,\ k_{1,[2,1]}=1,\ k_{1,[3,1]}=1,\ k_{1,[3,4]}=1;$
$k_{2,[1]}=1,\ k_{2,[5]}=1,\ k_{2,[9]}=1,\ k_{2,[12]}=1;$

All other family setup variables are zero. The Gantt chart of the optimal schedule is depicted in Fig. 4.

This MIP formulation can be used to solve problems containing only a few jobs and families. It becomes quite challenging to find optimal schedules for general problems. In practice, we may encounter problems with large numbers of jobs and families, so there is a need to develop heuristic methods which can quickly derive near-optimal solutions for medium- and large-size problems.

3.4. Heuristic methods

In Section 2.5, we explain that the one-piece processing and EDD rule are used as the current scheduling method. In order to effectively solve the emerged problems and improve the scheduling performance, we propose three heuristic methods.

3.4.1. Heuristic 1: FBEDD

We combine the full batch property with the EDD rule to develop a so-called Full Batch EDD (FBEDD) heuristic for minimizing the objective value. The detailed steps of the heuristic are given as follows

Step 1. Set $S=\phi$, $S_U=\{J_1 J_2,...,J_n\}$, $\mathcal{B}=\phi$, $i=1,2,...,b$ and $b=n/c$.
Step 2. Arrange jobs in S_U in non-decreasing order of d_j.
Let $i=1$.
Step 3. Take the first c jobs from S_U to form a batch B_i, i.e., $B_i=\{J_{[1]},...,J_{[c]}\}$, and add it to $\mathcal{B}$. Delete B_i from S_U.

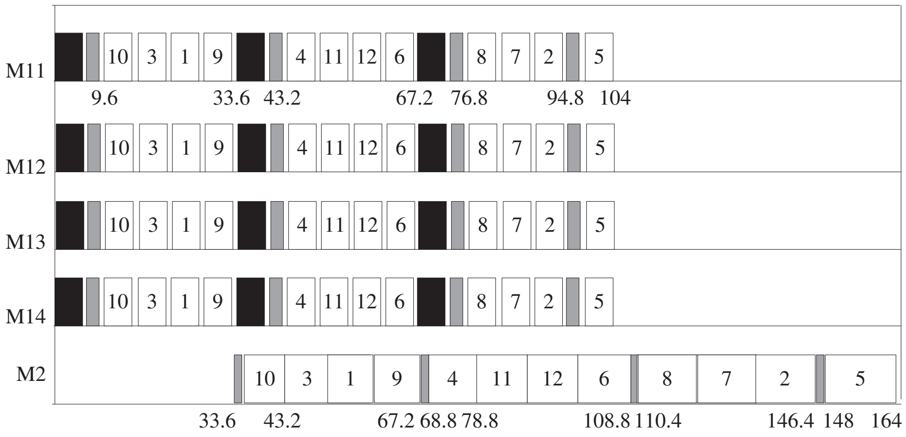

Fig. 4. Gantt chart of the optimal schedule for Example 1.

Step 4. If $S_U=\phi$, go to Step 5, otherwise $i=i+1$, go to Step 3.
Step 5. The final schedule is $S = \mathcal{B} = (B_1,\ldots,B_b)$.

3.4.2. *Heuristic 2: FBFS*

In the second heuristic, jobs are grouped into some family sets. We select each set of c jobs from the same family to form a full batch and place the remaining jobs coming from different families into the last batch, in which jobs are sequenced in Shortest Processing Time (SPT) order. The steps of the heuristic, named Full Batch Family Sorting (FBFS) heuristic, are given as follows, where # F^f denotes the number of jobs in F^f, the set of job family f.

Step 1. Set $S_1=\phi$, $S_2=\phi$, $S_U=\{J_1,J_2,\ldots,J_n\}$, $\mathcal{B}=\phi$, $\mathcal{B}'=\phi$ and $b=n/c$.
Step 2. Arrange the jobs in S_U in non-decreasing order of d_j.
Step 3. Sort the resulting sequence by job family to create a set. Let $f=1$.
Step 4. If # $F^f \geq c$, go to Step 7, otherwise go to Step 5.
Step 5. Set $i=1$, put F^f into B_i' as a partial batch and delete B_i' from S_U.
Step 6. If $S_U=\phi$, go to Step 9, otherwise let $f=f+1$ and go to Step 4.
Step 7. Set $i=1$, take the first c jobs from F^f as a full batch B_i^f and add it to $\mathcal{B}$. Delete B_i^f from S_U.
Step 8. If # $F^f \geq c$, go to Step 7, otherwise go to Step 5.
Step 9. Let $\mathcal{B} = B_1^1,\ldots,B_i^1,\ldots,B_1^4,\ldots,B_i^4$ and $\mathcal{B}'=B_1',\ldots,B_4'$. Rearrange the jobs in $\mathcal{B}'$ in non-decreasing order of p_{2j}.
Step 10. Leave job orders unchanged and re-index the batches in $\mathcal{B}$ and $\mathcal{B}'$. Let $\mathcal{B} = B_1,B_2,\ldots,B_{k-1}$ and $\mathcal{B}'=B_k,B_{k+1},\ldots,B_b$ where all batches are full.
Step 11. The final schedule is $S = \mathcal{B} \cup \mathcal{B}' = (B_1,B_2,\ldots,B_b) = (J_{[1]},J_{[2]},\ldots,J_{[n]})$.

We now elaborate on the above steps. In Steps 1 and 2, we construct an initial schedule, where jobs are sequenced in EDD order. In Step 3, the jobs are grouped into four families. In Steps 4–8, we take the first c jobs from each family to form a full batch and place the remaining jobs (in the partial batch) into the last batch. In Step 9, we rearrange jobs in each partial batch in SPT order to improve the objective function value. In Step 10, we re-index all batches without changing the sequences to make all batches to be full. In Step 11, we obtain the final schedule.

Example 2. Consider a 12-job problem with $p_{aj}=9$, $s_a=5.2$, $p_{1j}=24$, $s_b=6.4$ and $s_{f1}=3.2$, $s_{f2}=1.6$. Other data are the same as in Example 1.

Solution 1: Solving the problem by EDD, we obtain a final schedule with $C_{\max}=211$, $\sum C_j=1419.0$, $\sum T_j=66.2$ and $Z(S)=408.32$.

Solution 2: Solving the problem by FBEDD, we obtain a final schedule with $C_{\max}=178.4$, $\sum C_j=1344.4$, $\sum T_j=0$ and $Z(S)=375.92$.

Solution 3: Solving the problem by FBFS, we obtain a final schedule with $C_{\max}=164$, $\sum C_j=1163.2$, $\sum T_j=0$ and $Z(S)=331.04$.

3.4.3. *Heuristic 3: RFBFS*

When applying FBFS to large-size problems, the total setup time can be minimized. However, it gives priority to the jobs from the same family, regardless of the due dates. Thus, it may take a lot of time to process the batches with the same family, which leads some jobs to be processed earlier before they are needed and some other jobs to be completed tardy. To improve the drawback of FBFS, we combine the rolling horizon scheduling strategy (Fang and Xi, 1997) with FBFS heuristic to propose the Rolling FBFS (RFBFS) heuristic. RFBFS divides a large-size problem into several small-size problems by EDD order, and then performs FBFS repeatedly until all jobs are scheduled. Since RFBFS performs FBFS with small-size problem and shorter horizon, it can be applied to large-size problems. The steps of the heuristic are given as follows: where n_b denotes the number of batches which can be processed per month.

Step 1. Set $S_1=\phi$, $S_U=\{J_1,\ldots,J_n\}$. Set $\mathcal{B}_1 = \phi$, $\mathcal{B}_1'=\phi$, and $n=b \times c$.
Step 2. Arrange the jobs in S_U in non-decreasing order of d_j. Let $v=1$.
Step 3. Take the first $n_b/2$ jobs from S_U to perform FBFS, and create a partial schedule $S_1=\mathcal{B}_1 \cup \mathcal{B}_1'$. Then, $v=v+1$ and take the next $n_b/2$ jobs from S_U to perform FBFS, and create next partial schedule $S_2=\mathcal{B}_2 \cup \mathcal{B}_2'$.
Step 4. If $S_U=\phi$, go to Step 5; otherwise set $v=v+1$, go to Step 3.
Step 5. The final schedule is $S=S_1 \cup S_2 \cup \ldots \cup S_v$.

3.5. *Lower bound*

To evaluate the quality of the solution obtained from the heuristics, a lower bound is derived in this section. Since there are multiple objectives, the lower bound consists of three parts: (1) lower bound for $C_{\max}$, LB$_1$; (2) lower bound for $\sum C_j$, LB$_2$; (3) lower bound for $\sum T_j$, LB$_3$.

Lower bound for $C_{\max}$(LB$_1$)

Ahmadi et al. (1992) propose the LPT-Johnson rule for minimizing makespan on a batch-discrete system that processes multiple families of jobs. According to the rule, jobs are sequenced in non-increasing order of their processing times in stage 2. To obtain a reasonable lower bound, we combine the LPT-Johnson rule with the lower bound procedure reported by Azizoglu and Webster (2003). We include the batch setup time s_b and the family setup time s_{f1} into processing time of stage 1 and relax the family setup time in stage 2. The lower bound can then be obtained by the makespan value plus the minimal family setup time in stage 2 (i.e., $f_n \times s_{f2}$). Let $C_{2,[j]}$(LPT) denote the completion time of $J_{[j]}$ in the second stage, and f_n denote the number of families in all jobs. Then

$$C_{2,[1]}(\text{LPT}) = (s_b + s_{f1} + p_1) + p_{2,[1]}$$
$$C_{2,[j]}(\text{LPT}) = C_{2,[j-1]} + p_{2,[j]}, \quad j=2,\ldots,n$$
$$\text{LB}_1 = C_{2,[n]}(\text{LPT}) + f_n s_{f2}$$

Lower bound for $\sum C_j$(LB$_2$)

Ahmadi et al. (1992) also propose the Full Batch-SPT policy for minimizing total completion time on a batch-discrete system where no idle time is incurred after starting the processing in stage 2. According to the rule, jobs are sequenced in non-decreasing order of the processing time in stage 2. Denote by $C_{2,[j]}$(SPT) the completion time of job in stage 2. Then the lower bound can be obtained by the total completion time plus the minimal total family setup time in stage 2, where the family setup times are incurred only in the first job and the last f_n-1 jobs. That is,

$$C_{2,[1]}(\text{SPT}) = s_b + s_{f1} + p_1 + s_{f2} + p_{2,[1]}$$
$$C_{2,[j]}(\text{SPT}) = C_{2,[j-1]} + p_{2,[j]}, \quad j=2,\ldots,(n-f_n+1)$$
$$C_{2,[j]}(\text{SPT}) = C_{2,[j-1]} + s_{f2} + p_{2,[j]}, \quad j=(n-f_n+2),\ldots,n$$
$$\text{LB}_2 = \sum_{j=1}^{n} C_{2,[j]}(\text{SPT}) + (n + \sum_{k=1}^{f_n}(f_n-k))s_{f2}$$

Lower bound for $\sum T_j$(LB$_3$)

Baker and Martin (1974) shows that the SPT sequence minimizes total tardiness on a singe machine when all jobs are tardy. In this batch-discrete system, the batch processing times are identical, so the system can be considered as a singe machine when deriving a lower bound for total tardiness. Let $C_{2,[j]}$(SPT) and $d_{[j]}$(SPT), respectively, denote the completion time and due date of $J_{[j]}$ by SPT order. Then, a lower bound for total tardiness can be obtained by relaxing the constraint that all jobs are tardy. That is,

$$T_j = \max C_{2,[j]}(\text{SPT}) - d_{[j]}(\text{SPT}), 0, \quad j = 1,\ldots,n$$

$$\text{LB}_3 = \sum_{j=1}^{n} T_j$$

Combining the three individual lower bounds, the lower bound for the problem is

$$\text{LB} = \alpha \text{LB}_1 + \beta \text{LB}_2 + \gamma \text{LB}_3$$

To evaluate the quality of the solutions obtained from heuristics, we define the gaps between the heuristics and the lower bound as the Relative Error Rate (RER)

$$\text{RER} = \left(\frac{\text{Heuristic}_i - \text{LB}}{\text{LB}}\right) 100\%$$

where Heuristic$_i$ is the objective value of the respective heuristic.

Example 3. As an illustration of the lower bound, consider a 12-job problem with $p_{1j}=24$, $s_b=6.4$, $s_{f1}=3.2$, $s_{f2}=1.6$. Other data are the same as in Example 1.

The resulting lower bound with $(\alpha,\beta,\gamma)=(n/2,2,2)$ is

$$\text{LB} = 0.6 \times \text{LB}_1 + 0.2 \times \text{LB}_2 + 0.2 \times \text{LB}_3 = 0.6 \times 164 + 0.2 \times 1152 + 0.2 \times 0 = 328.8$$

The relative error rates for the three heuristics are calculated as

$$\text{RER}_{\text{EDD}} = \left(\frac{408.32 - 328.8}{328.8}\right) 100\% = 24.18\%$$

$$\text{RER}_{\text{FBEDD}} = \left(\frac{375.92 - 328.8}{328.8}\right) 100\% = 14.33\%$$

$$\text{RER}_{\text{FBFS}} = \left(\frac{331.04 - 328.8}{328.8}\right) 100\% = 0.68\%$$

The quality of solution obtained by FBFS is significantly better than EDD and FBEDD since RER$_{\text{FBFS}}$ has a much smaller value.

4. Computational experiments

In this section, computational experiments will be conducted to evaluate the performances of the MIP model and the proposed heuristics. Referring to the real data in Table 2, the assembly shop assembles about 24 jobs a month, so the experiments conducted consist of three parts: (1) small-size problems containing $n=12$ jobs for a planning horizon of 2 weeks; (2) medium-size problems containing $n=24$ and 60 jobs for planning horizons of 1 and 2.5 months, respectively; (3) large-size problems containing $n=100$, 200 and 300 jobs for planning horizons of 4 months, 8 months and 12 months, respectively. All heuristics are coded in MATLAB programming language and run on a PC with AMD Athlon 2.91 GHz CPU.

4.1. Evaluation for small-size problems

We randomly generated 10 instances ($N=10$) of small-size problems with 12 jobs and 4 families in which f_j is generated from a uniform distribution $DU(1, 4)$ and d_j from $DU(100,180)$. To evaluate the performance of the proposed heuristics, we consider two measures: solution quality and running time (Hamta et al., 2012). To examine the solution quality for a heuristic, we calculate the average relative percentage deviation (ARPD) as follows (Laha and Sarin, 2009)

$$\text{ARPD} = \frac{100}{N} \sum_{i=1}^{N} \left(\frac{\text{Heuristic}_i - \text{Optimal}_i}{\text{Optimal}_i}\right),$$

where Heuristic$_i$ is the objective value of the ith instance obtained by a heuristic, Optimal$_i$ is the MIP objective value of the ith instance. We compute objective values and ARPD for three heuristics EDD, FBEDD and FBFS with weights of performance measures $(\alpha,\beta,\gamma)=(n/2,2,2)=(12/2,2,2)=(0.6,0.2,0.2)$ in Table 3. It can be observed from the last row that the ARPD of EDD is 23.38% (ranging from 14.44% to 30.95%) and the ARPD of FBEDD is 10.49% (ranging from 4.75% to 14.90%), while the ARPD of FBFS is only 0.45% (ranging from 0% to 1.73%), which is much smaller. Thus, the performance of FBFS is significantly better than the two others. As for the running time, MIP requires an average of 34 s (ranging from 8 to 123 s) for each run, while FBEDD and FBFS both take less than 0.3 s. Therefore, FBFS is a good method for small-size problems.

4.2. Evaluation for medium- and large-size problems

Taking the rush orders into account, the medium- and large-size problems are designed to have 80% MTO (Make-To-Order) jobs and 20% MTS (Make-To-Stock) jobs. We set the shortest delivery date D_j for MTO orders at least 15 days (15×8 h$=120$ h), and the longest delivery date as one year (365×8 h$=3000$ h). So, the delivery date interval is defined as [120 h, 3000 h]. The due dates d_j are determined based on the delivery dates and the limited capacity. The normal due dates ($d_j=D_j$) are set based on the normal capacity, while the tighter due dates ($d_j=D_j-40$) and looser due dates ($d_j=D_j+40$) are set based on the tight and loose capacity, respectively.

Since the problem is NP-hard, we cannot find the optimal solutions for medium- and large-size problems within a reasonable time. To evaluate the solution quality of the heuristics, we compare the gaps between the heuristic solutions and the lower bound using the Relative Error Rate (RER), as defined above. Furthermore, to compare the performance of the heuristics with the current method, we define the Relative Improvement Rate (RIR) as the performance measure (Mokhtari et al., 2010)

$$\text{RIR}(\%) = \frac{100}{N} \sum_{i=1}^{N} \left(\frac{\text{Heuristic}_i - \text{Best}_i}{\text{Best}_i}\right),$$

where Heuristic$_i$ is the solution value of the ith instance obtained by a heuristic and Best$_i$ is the best solution value of the instance.

For medium- and large-size problems, 15 problem sizes with a combination of 5 job sizes ($n=24$, 60, 100, 200 and 300) and

Table 3
Comparison of MIP and three heuristics for ARPD.

No	MIP		EDD		FBEDD		FBFS	
	Z	time(s)	Z	ARPD	Z	ARPD	Z	ARPD
1	331.04	10	423.64	27.97	375.92	13.56	331.04	0
2	374.56	15	435.88	16.37	396.88	5.96	376.48	0.51
3	338.72	21	425.64	25.66	380.08	12.21	344.48	1.70
4	332.40	123	399.48	20.18	354.32	6.59	338.16	1.73
5	377.28	35	435.84	15.52	395.20	4.75	377.28	0
6	342.64	47	441.2	28.76	393.68	14.90	342.64	0
7	316.64	24	414.64	30.95	349.64	10.42	316.64	0
8	345.36	22	443.28	28.35	391.48	13.35	345.36	0
9	359.04	35	451.04	25.62	399.44	11.25	359.04	0
10	386.72	8	442.56	14.44	432.80	11.92	389.04	0.60
Avg.	350.44	34	431.32	23.38	386.94	10.49	352.02	0.45

3 levels of due dates (Tighter (T), Normal (N) and Looser (L)) are tested. Each problem size consists of 10 instances, in which f_j is generated from $DU(1,4)$ and D_j is generated from $DU(120,360)$, $DU(120,720)$, $DU(120, 1120)$, $DU(120, 2120)$ and $DU(120, 3000)$ for the 5 job sizes, respectively. We compute the lower bound and objective values of the three heuristics for problem $n=24$, $d_j=D_j$ with weights $(\alpha,\beta,\gamma)=(n/2,2,2)=(24/2,2,2)=(0.75, 0.125, 0.125)$ and summarize RER and RIR in Table 4. The last row of the table shows that EDD and FBEDD have an average RER of 35.95% and 13.30%, respectively, while RFBFS has an average RER of 5.14%. It also shows that RFBFS outperforms EDD and FBEDD by having an average RIR of 29.31% and 7.76%, respectively. Thus, RFBFS is much better than the others in terms of the solution quality and performance improvement.

In the same manner, we compute the objective values, RER and RIR with weights $(\alpha,\beta,\gamma)=(n/2,2,2)$ for all the combinations. The results of the average values are summarized in Table 5, which shows that EDD and FBEDD have the average RER of 35.95%–46.47% and 10.92%–17.16%, respectively, while RFBFS obtains better average RER of 3.48%–7.42%. Also, it can be observed that the average RIR of RFBFS is consistently better than EDD and FBEDD by having average RIR of 27.15%–37.93% and 6.29%–9.79%, respectively. Thus, it can be concluded that RFBFS outperforms EDD and FBEDD. These imply that the properties of full batching and family sorting have positive effect on this problem and the effect is strengthened as the problem size is increased. Moreover, it is noted that in almost all the problems the tighter the due date, the better RER and RIR are obtained. This means that the strict control of due dates has a positive effect on the scheduling performance.

In the previous experiments, it is assumed for simplicity that there are only four job families ($f=4$). To verify that RFBFS can be applied in other situations, more experiments are conducted. We modify the case $n=24$ where the families of jobs 18, 19, 20 were replaced by $f_j=5$ for $f=5$, and the families of jobs 15, 16, 17 were replaced by $f_j=6$ for $f=6$. The results are summarized in Table 6, which shows that EDD and FBEDD solutions have an average RER of 35.95%–43.49% and 10.92%–18.02%, respectively, while RFBFS solution has an average RER of 3.48%–9.54%. Also, it can be observed that RFBFS has an average RIR of 7.19%–8.68% over FBEDD and 27.15%–34.29% over EDD. Thus, it can be concluded that RFBFS outperforms EDD and FBEDD in terms of the solution

Table 4
Comparison of three heuristics for 10 instances of problem $n=24$, $d_j=D_j$.

No	LB	Objective value (Z)			RER (%)			RIR (%)	
		(1) EDD	(2) FBEDD	(3) RFBFS	$\frac{(1)-LB}{LB}$	$\frac{(2)-LB}{LB}$	$\frac{(3)-LB}{LB}$	$\frac{(1)-(3)}{(3)}$	$\frac{(2)-(3)}{(3)}$
1	661.5	953.4	748.9	698.9	44.13	13.22	5.66	36.41	7.15
2	723.3	1047.6	824.2	744.4	44.84	13.95	2.92	40.73	10.72
3	723.2	998.2	825.3	765.8	38.02	14.11	5.88	30.35	7.77
4	722.7	956.5	811.6	743.5	32.36	12.31	2.88	28.66	9.17
5	762.2	1029.9	898.9	808.5	35.12	17.94	6.07	27.38	11.18
6	661.2	918.9	733.4	683.9	38.98	10.92	3.43	34.37	7.24
7	761.1	963.3	846.8	791.3	26.58	11.26	3.97	21.74	7.01
8	732.8	997.7	850.3	790.6	36.15	16.03	7.89	26.19	7.54
9	748.4	934.0	830.9	791.3	24.80	11.03	5.73	18.03	5.01
10	663.9	934.8	742.0	709.9	40.80	11.76	6.92	31.68	4.53
Avg.	716.0	973.4	811.2	752.8	35.95	13.30	5.14	29.31	7.76

Table 5
Comparison of three heuristics for all combinations.

n	d_j	LB	Objective value (Z)			RER (%)			RIR (%)	
			(1) EDD	(2) FBEDD	(3) RFBFS	$\frac{(1)-LB}{LB}$	$\frac{(2)-LB}{LB}$	$\frac{(3)-LB}{LB}$	$\frac{(1)-(3)}{(3)}$	$\frac{(2)-(3)}{(3)}$
24	T	743	1017	824	769	36.77	10.92	3.48	32.17	7.19
	N	716	973	811	753	35.95	13.30	5.14	29.31	7.76
	L	701	957	811	753	36.59	15.78	7.42	27.15	7.78
60	T	2036	2892	2339	2156	42.08	14.91	5.92	34.14	8.49
	N	1977	2802	2268	2094	41.71	14.70	5.91	33.80	8.30
	L	1931	2735	2221	2054	41.65	15.04	6.38	33.15	8.14
100	T	3537	5170	4144	3774	46.18	17.16	6.71	36.99	9.79
	N	3468	5050	4045	3702	45.61	16.64	6.76	36.40	9.26
	L	3403	4939	3939	3633	45.16	15.77	6.79	35.93	8.41
200	T	7208	10,552	8301	7662	46.39	15.17	6.30	37.71	8.34
	N	7141	10,412	8169	7592	45.80	14.39	6.31	37.14	7.59
	L	7077	10,278	8044	7524	45.22	13.66	6.31	36.59	6.91
300	T	10,817	15,844	12,338	11,487	46.47	14.06	6.19	37.93	7.41
	N	10,753	15,697	12,196	11,418	45.98	13.42	6.19	37.48	6.81
	L	10,690	15,555	12,060	11,350	45.51	12.82	6.17	37.05	6.26

Table 6
Comparison of three heuristics with different family numbers for $n=24$.

f	d_j	LB	Objective value (Z)			RER (%)			RIR (%)	
			(1) EDD	(2) FBEDD	(3) RFBFS	$\frac{(1)-\text{LB}}{\text{LB}}$	$\frac{(2)-\text{LB}}{\text{LB}}$	$\frac{(3)-\text{LB}}{\text{LB}}$	$\frac{(1)-(3)}{(3)}$	$\frac{(2)-(3)}{(3)}$
4	T	743	1017	824	769	36.77	10.92	3.48	32.17	7.19
	N	716	973	811	753	35.95	13.30	5.14	29.31	7.76
	L	701	957	811	753	36.59	15.78	7.42	27.15	7.78
5	T	740	1040	844	780	40.48	14.00	5.34	33.36	8.22
	N	714	985	819	760	37.93	14.71	6.30	29.76	7.92
	L	698	967	813	756	38.55	16.58	8.38	27.84	7.56
6	T	737	1057	855	787	43.49	16.13	6.85	34.29	8.68
	N	713	1000	829	769	40.19	16.31	7.81	30.04	7.88
	L	696	978	821	762	40.53	18.02	9.54	28.29	7.74

Table 7
Performance measures for the shop floor.

Period (month)	Number of jobs	Average starvation time (h)	Average flow time (h)	On-time completion rate
0	24	2.67	228	0.15
1	12	1.17	128	0.50
2	12	1.00	121	0.60
3	16	0.63	160	0.65
4	16	0.75	158	0.78
5	24	0.75	194	0.82
6	24	0.71	188	0.86

quality and improvement measure and, moreover, both measures slightly increase as the number of job families is increased.

4.3. Real-life performance evaluation

To further validate the performance of the proposed scheduling method, we provide three performance criteria, according to the three problems stated in Section 2.5, including starvation time, flow time and on-time completion rate. We collect the real data of the shop floor from January to June 2009 and compare the performance before and after the implementation of the proposed method. The results are given in Table 7 and Fig. 5, where period 0 represents the measures before the use of the proposed method. It can be observed that the starvation time has been greatly reduced because of the batch processing. Also, the flow time and the on-time completion rate have been significantly improved since job dividing and family scheduling are performed.

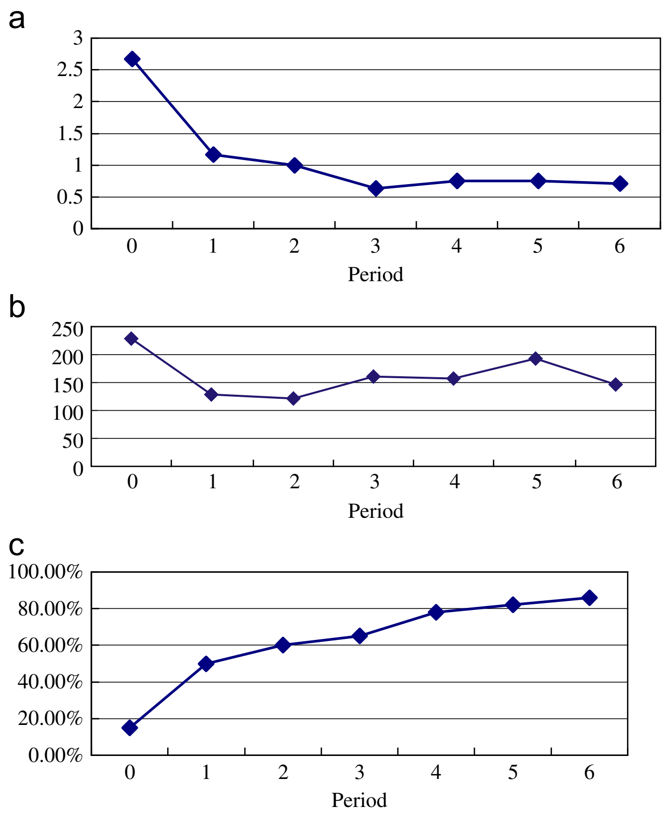

Fig. 5. Performance measures for the shop floor. (a) Average starvation time, (b) average flow time and (c) on-time completion rate.

5. Conclusions and future research

In this paper, we have considered the scheduling problem for an assembly shop in a machinery factory. We have proposed the strategies of job dividing and batch processing which make the shop more efficient. We have also adopted an 80/20 of mixed MTO/MTS policy to deal with the insertion of rush orders. The MIP model and three heuristics, FBEDD, FBFS and RFBFS, have been developed to solve the assembly shop scheduling problem with the objective of minimizing the weighted sum of makespan, total completion time, and total tardiness. The computational experiments show that FBFS outperforms the others for solving small-size problems while RFBFS solution has a good quality for solving medium- or large-size problems.

We have demonstrated that RFBFS can be applied to a wide variety of situations in the two-stage assembly shop. A useful extension is to modify the heuristics to multi-stage assembly factories. Besides, jobs in most factories are scheduled based on MRP due dates with rolling horizon, and hence how to perform a periodic scheduling in an assembly factory is an important issue for future research.

References

Ahmadi, J.H., Ahmadi, R.H., Dasu, S., Tang, C.S., 1992. Batching and scheduling jobs on batch and discrete processors. Oper. Res. 39, 750–763.
Allahverdi, A., Gupta, J.N.D., Aldowaisan, T., 1999. A review of scheduling research involving setup considerations. Omega 27, 219–239.

Amorim, P., Günther, H.-O., Almada-Lobo, B., 2012. Multi-objective integrated production and distribution planning of perishable products. Int. J. Prod. Econ. 138, 89–101.

Azizoglu, M., Webster, S., 2003. Scheduling parallel machines to minimize weighted flow time with family set-up times. Int. J. Prod. Res. 41, 1199–1215.

Baker, K.R., Martin, J.B., 1974. An experimental comparison of solution algorithms for the single-machine tardiness problem. Nav. Res. Logistics Quarterly 21, 187–199.

Baker, K.R., 1999. Heuristic procedures for scheduling job families with setups and due dates. Naval Research Logistics 46, 978–991.

Baker, K.R., Magazine, M.J., 2000. Minimizing maximum lateness with job families. Eur. Oper. Res. 127, 126–139.

Bukchin, J., Tzur, M., Jaffe, M., 2002. Lot splitting to minimize average flow-time in a two-machine flow-shop. IIE Trans. 34, 953–970.

Du, J., Leung, J., 1990. Minimizing total tardiness on one machine is NP-hard. Math. Oper. Res. 15, 483–495.

Eren, T., 2007. A multicriteria flow shop scheduling problem with setup times. J. Mater. Process. Technol. 186, 60–65.

Gong, H., Tang, L., Duin, C.W., 2010. A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times. Comput. Oper. Res. 37, 960–969.

Fang, J., Xi, Y., 1997. A rolling horizon job shop rescheduling strategy in the dynamic environment. Int. J. Adv. Manuf. Technol. 13, 227–232.

Framinan, J.M., Leisten, R., Ruiz-Usano, R., 2002. Efficient heuristics for flow shop sequencing with the objectives of makespan and flow time minimization. Eur. J. Oper. Res. 141, 559–569.

Hamta, N., Fatemi Ghomi, S.M.T., Jolai, F., Akbarpour Shirazi, M., 2012. A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. International Journal of Production Economics. doi:10.1016/j.ijpe.2012.03.013.

Hillier, F.S., Lieberman, G.J., 2001. Introduction to Operations Research, seventh ed. McGraw-Hill, New York, pp 576–590.

Hoogeveen, J.A., van de Velde, S.L., 1998. Scheduling by positional completion times: analysis of a two-stage with a batching machine. Math. Programming 82, 273–289.

Kim, B., Kim, S., 2002. Application of genetic algorithms for scheduling batch-discrete production system. Prod. Plan. Control 13, 155–165.

Koulamas, C., Kyparisis, G.J., 2007. A note on the two-stage assembly flow shop scheduling problem with uniform parallel machines. Eur. J. Oper. Res. 182, 945–951.

Laha, D., Sarin, S.C., 2009. A heuristic to minimize total flow time in permutation flow shop. Omega 37, 734–739.

Lee, C.Y., Uzsoy, R., Martin-Vega, L.A., 1992. Efficient algorithms for scheduling semi-conductor burn-in operations. Oper. Res. 40, 764–775.

Li, S.G., Rong, Y.L., 2009. The reliable design of one-piece flow production system using fuzzy ant colony optimization. Comput. Oper. Res. 36, 1656–1663.

Liao, C.J., Liao, L.M., 2008. Improved MILP models for two-machine flow shop with batch processing machines. Math. Comput. Modeling 48, 1254–1264.

Lin, B.M.T., Cheng, T.C.E., 2002. Fabrication and assembly scheduling in a two machine flow shop. IIE Trans. 34, 1015–1020.

Liu, L.L., Ng, C.T., Cheng, T.C.E., 2009. Bicriterion scheduling with equal processing times on a batch processing machine. Comput. Oper. Res. 36, 110–118.

Liu, Z., Yu, W., 2000. Scheduling one batch processor subject to job release dates. Discrete Appl. Math. 105, 129–136.

Logendran, R., Carson, S., Hanson, E., 2005. Group scheduling in flexible flow shops. Int. J. Prod. Econ. 96, 143–155.

Mokhtari, H., Abadi, I.N.K., Cheraghalikhani, A., 2010. A multi-objective flow shop scheduling with resource-dependent processing times: trade-off between makespan and cost of resources. Int. J. Prod. Res. iFirst, 1–25.

Neale, J.J., Duenyas, I., 2000. Control of manufacturing networks which contain a batch processing machine. IIE Trans. 32, 1027–1041.

Nearchou, A.C., 2011. Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization. Int. J. Prod. Econ. 129, 242–250.

Potts, C.N., Kovalyov, M.Y., 2000. Scheduling with batching: a review. Eur. J. Oper. Res. 120, 228–249.

Potts, C.N., Strusevich, V.A., Tautenhahn, T., 2001. Scheduling batches with simultaneous job processing for two-machine shop problems. J. Scheduling 4, 25–51.

Schaller, J., 2007. Scheduling on a single machine with family setups to minimize total tardiness. Int. J. Prod. Econ. 105, 329–344.

Schaller, J.E., Gupta, J.N.D., 2008. Single machine scheduling with family setups to minimize total earliness and tardiness. Eur. J. Oper. Res. 187, 1050–1068.

Sung, C.S., Kim, H.A., 2008. A two-stage multiple-machine assembly scheduling problem for minimizing sum of completion times. Int. J. Prod. Econ. 113, 1038–1048.

Taboada, H.A., Coit, D.W., 2008. Multi-objective scheduling problems: determination of pruned Pareto sets. IIE Trans. 40, 552–564.

Yokoyama, M., 2001. Hybrid flow shop scheduling with assembly operations. Int. J. Prod. Econ. 73, 103–116.

Webster, S., Baker, K.R., 1995. Scheduling groups of jobs on a singe machine. Oper. Res. 43, 692–703.