A Heuristic Approach for Tow-machine No-wait Flowshop Scheduling with Due Dates and Class Setups

Xiuli Wang[a] T. C. E. Cheng[b, #]

[a]College of Electrical Engineering
Zhejiang University, P. R. China

[b]Department of Logistics
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
E-mail: LGTcheng@polyu.edu.hk

Abstract

This paper studies the two-machine flowshop scheduling problem with class setups in a no-wait processing environment to minimize the maximum lateness. The jobs are divided into job classes and a setup is required at the initial processing of a job or between the processing of jobs of different classes. In a no-wait environment, a job must be processed from start to finish without interruptions on a machine or between the machines. A batch is a maximal subset of consecutively processed jobs of the same class. Several properties concerning the beneficial merging of batches and some dominance rules that improve the objective function are derived. Since the problem is NP-hard, a heuristic is proposed and evaluated computationally. The numerical results demonstrate that the heuristic can produce near-optimal solutions quickly for realistic-sized problems.

Keywords: flowshop scheduling, job class setups, no-wait, heuristic, maximum lateness.

[#]Corresponding author.

1. Introduction

An important kind of flowshop scheduling problem is characterized by a no-wait environment. In a no-wait environment, a job must be processed from start to finish without interruptions on a machine or between machines. The no-wait flowshop scheduling problem arises in many industries. Examples include the metal, plastic and chemical processing industries, as well as advanced manufacturing, such as just-in-time production. The no-wait scheduling problem has attracted the attention of many researchers. A detailed survey of the research and applications on this topic has been given by Hall and Sriskandarajah [1].

Recently, many basic scheduling problems have been extended to include considerations of job classes. In a scheduling environment with job classes, the jobs are divided into several classes according to the similarities of their processing requirements. The processing of a job does not require a setup if it follows a job from the same class, but it requires a known class setup if it is the first job to be processed on a machine or it follows a job from another class. A batch is a maximal subset of consecutively processed jobs of the same class. When the number of batches is fixed, the scheduling decisions are made at two levels [2]. At the first level, the sequence of the jobs within each batch is decided, and at the second level, the sequence of the batches is determined. Sekiguchi [3] proposed a method for solving the two-machine flowshop scheduling problem under the series-parallel precedence constraints to minimize the makespan. Yang and Chern [4] solved the two-machine group scheduling problem to minimize the makespan, in which there is a transportation time between the two machines. Baker [5] reviewed research on the two-machine flowshop scheduling problem and provided a generalized framework for the class scheduling problem to minimize the makespan. When the job classes can be split or the number of batches is not fixed, the class scheduling problem becomes very difficult to deal with. Pan *et al*. [6] presented a heuristic approach for the single-machine batch scheduling to minimize the maximum lateness. Webster and Baker [7] reviewed work on the single-machine batch scheduling problem and presented some properties for the minimization of the total weighted flowtime and the maximum lateness. Monma and Potts [8], and Potts and Van Wassenhove [9] analyzed the computational complexity of the batch scheduling problem for various objective functions. Potts and Kovalyov [10]

summarized the various dynamic programming algorithms for solving variants of the batch scheduling problem.

In contrast to the existence of many research results on either the no-wait flowshop scheduling problem or on scheduling problems with class setups, there have been few attempts to study scheduling problems that involve both aspects although they may be applicable in actual practice. In the following, we give an example of rolling steel tubes. Steel tubes are divided into various job classes according to their outer and inner diameters. The production process consists of two stages. First, by setting different mechanical frames, steel pillars are rolled as steel tubes with different outer-diameters. Then, steel tubes with different inner-diameters are produced through setting different core-sticks. The process of rolling a steel tube cannot be interrupted since its temperature must be preserved until finished.

In this paper we consider the two-machine flowshop scheduling problem with job classes in a no-wait processing environment to minimize the maximum lateness. Since no-wait scheduling with several machines is at least as difficult as that with a single machine, whatever the configuration of the machines [1], and the single-machine job class scheduling problem to minimize the maximum lateness is NP-hard when the number of batches is arbitrary [8], the scheduling problem under study is NP-hard. The paper is organized as follows. In Section 2, we introduce the notation and present some definitions. We derive some theoretical results in Section 3. In Section 4, we design a heuristic algorithm for the problem, and in Section 5 we evaluate the performance of the heuristic computationally and present the numerical results. Finally, some conclusions are given in Section 6.

2. Notation and definitions

A set of n jobs are given to be scheduled in a two-machine no-wait flowshop to minimize the maximum lateness. The jobs are divided into c job classes. We define job (i, j) as the job numbered j in class i. Let a_{ij} (>0)and b_{ij} (>0) denote the processing times of job (i, j) on machines 1 and 2, respectively, and d_{ij} the due date of job (i, j).

In a schedule that includes r batches ($r \geq c$), let $B_k(i)$ denote the ith batch that belongs to class k, and $[i, j]$ the jth job in the ith batch, where $i = 1, 2, \cdots, r$, $j = 1, 2, \cdots, \beta_i$. Then, we

have the following notation.

$a_{[i,j]}$: processing time of job $[i,j]$ on machine 1;

$b_{[i,j]}$: processing time of job $[i,j]$ on machine 2;

$d_{[i,j]}$: due date of job $[i,j]$;

$s_{[i],k}$: setup time of the ith batch on machine k, $k = 1, 2$;

$C_{[i,j]}$: completion time of job $[i,j]$;

$L_{[i,j]}$: lateness of job $[i,j]$;

$C_k(i)$: completion time of batch $B_k(i)$;

$L_k(i)$: maximum lateness of $B_k(i)$;

$C_{[i]}$: completion time of the ith batch;

L_{max}: maximum lateness of a schedule.

In a no-wait environment, we have

$$C_{[i,j]} = \sum_{p=1}^{i}(s_{[p],2} + \max\{0, s_{[p],1} + a_{[p,1]} - s_{[p],2} - b_{[p-1,\beta_{p-1}]}\} + b_{[p,1]})$$

$$+ \sum_{k=1}^{i-1}(\sum_{p=2}^{\beta_k}\max\{0, a_{[k,p]} - b_{[k,p-1]}\} + \sum_{p=2}^{\beta_k} b_{[k,p]}) + \sum_{p=2}^{j}(\max\{0, a_{[i,p]} - b_{[i,p-1]}\} + b_{[i,p]}),$$

where we assume that $b_{[0,\beta_0]} = 0$, and

$$L_{[i,j]} = C_{[i,j]} - d_{[i,j]}.$$

We define a batch due date, which allows us to treat a batch as a composite job in a schedule.

Definition 1 For a batch that is in the ith position of a schedule, define a batch due date, δ_i, as

$$\delta_i = \min\{\delta_{i1}, \delta_{i2}, \cdots, \delta_{i\beta_i}\},$$

where

$$\delta_{ij} = d_{[i,j]} + \sum_{p=j+1}^{\beta_i}\max\{0, a_{[i,p]} - b_{[i,p-1]}\} + \sum_{p=j+1}^{\beta_i} b_{[i,p]}, \text{ for } j = 1, 2, \cdots, \beta_i.$$

From Definition 1, we see that δ_i is independent of the time at which the batch starts to be processed. To extend the idea of the due date of a batch to a series of consecutive batches, we give the following definition.

Definition 2 The due date of the *i*th through *j*th batches is defined as

$$\Delta_{ij} = \min_{i\le k\le j}\{\delta_k + \sum_{q=k+1}^{j}(s_{[q],2} + \max\{0, s_{[q],1} + a_{[q,1]} - s_{[q],2} - b_{[q-1,\beta_{q-1}]}\}$$

$$+ \sum_{p=2}^{\beta_q}\max\{0, a_{[q,p]} - b_{[q,p-1]}\} + \sum_{p=1}^{\beta_q} b_{[q,p]})\}\ .$$

From the definition of Δ_{ij}, we notice that the value of Δ_{ij} is independent of the time at which the batches begin to be processed.

From Definition 1, we have

$$C_{[i]} - \delta_i = C_{[i,\beta_i]} - \min\{\delta_{i1}, \delta_{i2}, \cdots, \delta_{i\beta_i}\}$$

$$= \max\{C_{[i,\beta_i]} - \delta_{i1}, C_{[i,\beta_i]} - \delta_{i2}, \cdots, C_{[i,\beta_i]} - \delta_{i\beta_i}\}$$

$$= \max\{C_{[i,\beta_i]} - (\sum_{p=2}^{\beta_i}\max\{0, a_{[i,p]} - b_{[i,p-1]}\} + \sum_{p=2}^{\beta_i} b_{[i,p]}) - d_{[i,1]},$$

$$C_{[i,\beta_i]} - (\sum_{p=3}^{\beta_i}\max\{0, a_{[i,p]} - b_{[i,p-1]}\} + \sum_{p=3}^{\beta_i} b_{[i,p]}) - d_{[i,2]}, \cdots, C_{[i,\beta_i]} - d_{[i,\beta_i]}\}$$

$$= \max\{C_{[i,1]} - d_{[i,1]}, C_{[i,2]} - d_{[i,2]}, \cdots, C_{[i,\beta_i]} - d_{[i,\beta_i]}\},$$

which means that the lateness of a batch is equal to the maximum lateness of the jobs in this batch. Definition 2 of a serial batches has a similar meaning as that of Definition 1.

3. Theoretical results

An initial schedule can be constructed by sequencing all the jobs in increasing order of the job due dates. The maximum lateness of this schedule can be computed by adding the setup times on machine 1 and 2 whenever there is a change of class between two consecutive jobs. According to Definition 1, it is obvious that the batches are sequenced in increasing order

of the batch due dates in this initial schedule. We denote the schedule in which the batches are sequenced in increasing order of their due dates as a BEDD schedule. Apparently, this initial schedule may be improved by combining some batches of the same class to decrease the number of setups.

A forward merge is defined as a forward movement of a batch to be combined with another batch of the same class sequenced before it, and a backward merge is a backward movement of a batch to be combined with another batch of the same class sequenced after it. The merged positions are defined as the positions of the batches that are changed because of the merge.

A forward or a backward merge may decrease the maximum lateness of a schedule. Therefore, properties are developed to evaluate the benefits resulting from such merges.

In the following, although the two merged batches actually become a batch, for the convenience of developing the theoretical results, we still denote it as two adjacent batches.

Property 1 In a BEDD schedule, for two batches of class i, $B_i(k)$ and $B_i(l)$,

i) a forward merge of $B_i(l)$ will cause the maximum lateness of the (k+1)st to lth merged positions not to occur in $B_i(k+1)$ in the new schedule;

ii) a backward merge of $B_i(k)$ will cause the maximum lateness of the kth to $(l-1)$ st merged position to occur in $B_i(l-1)$ in the new schedule.

Proof i) Consider the two schedules σ and σ_1 in Fig.1 (a), where σ is a BEDD schedule and σ_1 is the same as σ except that $B_i(l)$ is merged forward with $B_i(k)$. Since σ is a BEDD schedule, the due date of $B_i(l)$ is no less than that of $B_p(u)$, for $k < u < l$ and $p \neq i$. The forward merge of $B_i(l)$ will cause the lateness of $B_i(k+1)$ to be less than that of $B_p(u+1)$ in σ_1. Therefore, the maximum lateness of the batches between the (k+1)st and the lth positions will not occur in $B_i(k+1)$ in σ_1.

ii) Consider the two schedules σ and σ_2 in Fig.1 (b). Since $B_i(k)$ has the earliest due date

among all the batches from positions k to l in σ, after a backward merge, $B_i(k)$ will have the maximum lateness of the batches between the kth and the $(l-1)$ st positions in σ_2.. □

Let $B_i(k)$ and $B_i(l)$, where $k < (l-1)$, be two batches of class i in a BEDD schedule. Define L_f as be the maximum lateness of the batches in the (k+1)st to lth merged positions if $B_i(l)$ is merged forward. Also let L_b be the maximum lateness of the batches in the kth to $(l-1)$ st merged positions if $B_i(k)$ is merged backward. We assume that the maximum lateness of a BEDD schedule, L_{max}, occurs in a batch sequenced after the lth position. We denote the maximum lateness of the new schedule σ_1 as $L_{\max}(\sigma_1)$ if $B_i(l)$ is merged forward, and the maximum lateness of the new schedule σ_2 as $L_{\max}(\sigma_2)$ if $B_i(k)$ is merged backward. For the generated schedules σ_1 and σ_2, we consider the following four cases:

1) The batches in the (k–1)st and (k+1)st positions of the schedule are not of the same class, nor are the batches in the (l–1)st and (l+1)st positions. Then, we have

$$L_f = \max\{0, a_{[l,1]} - b_{[k,\beta_k]}\} + \sum_{p=2}^{\beta_l} \max\{0, a_{[l,p]} - b_{[l,p-1]}\}$$

$$+ \sum_{p=1}^{\beta_l} b_{[l,p]} + \max\{0, s_{[k+1],1} + a_{[k+1,1]} - s_{[k+1],2} - b_{[l,\beta_l]}\}$$

$$- \max\{0, s_{[k+1],1} + a_{[k+1,1]} - s_{[k+1],2} - b_{[k,\beta_k]}\} + \max\{L_q(u) \mid k < u < l, q \neq i\},$$

$$L_b = s_{[k+1],2} + \max\{0, s_{[k+1],1} + a_{[k+1,1]} - s_{[k+1],2} - b_{[k-1,\beta_{k-1}]}\} + \sum_{p=2}^{\beta_{k+1}} \max\{0, a_{[k+1,p]} - b_{[k+1,p-1]}\}$$

$$+ \sum_{p=1}^{\beta_{k+1}} b_{[k+1,p]} + \sum_{q=k+2}^{l-1} (s_{[q],2} + \max\{0, s_{[q],1} + a_{[q,1]} - s_{[q],2} - b_{[q-1,\beta_{q-1}]}\}$$

$$+ \sum_{p=2}^{\beta_q} \max\{0, a_{[q,p]} - b_{[q,p-1]}\} + \sum_{p=1}^{\beta_q} b_{[q,p]}) + \max\{0, s_{[k],1} + a_{[k,1]} - s_{[k],2} - b_{[l-1,\beta_{l-1}]}\}$$

$$- \max\{0, s_{[k],1} + a_{[k,1]} - s_{[k],2} - b_{[k-1,\beta_{k-1}]}\} + L_i(k),$$

$$L_{\max}(\sigma_1) = \max\{0, a_{[l,1]} - b_{[k,\beta_k]}\} + \max\{0, s_{[k+1],1} + a_{[k+1,1]} - s_{[k+1],2} - b_{[l,\beta_l]}\}$$

$$+\max\{0,\, s_{[l+1],1}+a_{[l+1,1]}-s_{[l+1],2}-b_{[l-1,\,\beta_{l-1}]}\}-\max\{s_{[l],2},\, s_{[l],1}+a_{[l,1]}-b_{[l-1,\,\beta_{l-1}]}\}$$

$$-\max\{0,\, s_{[k+1],1}+a_{[k+1,1]}-s_{[k+1],2}-b_{[k,\,\beta_k]}\}-\max\{0,\, s_{[l+1],1}$$

$$+a_{[l+1,1]}-s_{[l+1],2}-b_{[l,\,\beta_l]}\}+L_{\max}, \qquad (1)$$

$$L_{\max}(\sigma_2)=\max\{0,\, s_{[k+1],1}+a_{[k+1,1]}-s_{[k+1],2}-b_{[k-1,\,\beta_{k-1}]}\}+\max\{0,\, s_{[k],1}+a_{[k,1]}-s_{[k],2}$$

$$-b_{[l-1,\,\beta_{l-1}]}\}+\max\{0,\, a_{[l,1]}-b_{[k,\,\beta_k]}\}-\max\{\, s_{[k],2},\, s_{[k],1}+a_{[k,1]}-b_{[k-1,\,\beta_{k-1}]}\}$$

$$-\max\{0,\, s_{[k+1],1}+a_{[k+1,1]}-s_{[k+1],2}-b_{[k,\,\beta_k]}\}-\max\{0,\, s_{[l],1}+a_{[l,1]}-s_{[l],2}$$

$$-b_{[l-1,\,\beta_{l-1}]}\}+L_{\max}. \qquad (2)$$

2) The batches in the (k–1)st and (k+1)st positions of the schedule are of the same class, but the batches in the (l–1)st and (l+1)st positions are not. Then, L_f and $L_{max}(\sigma_1)$ are the same as those in 1). For L_b and $L_{max}(\sigma_2)$, we have

$$L_b=\max\{0,\, a_{[k+1,1]}-b_{[k-1,\,\beta_{k-1}]}\}+\sum_{p=2}^{\beta_{k+1}}\max\{0,\, a_{[k+1,\,p]}-b_{[k+1,\,p-1]}\}+\sum_{p=1}^{\beta_{k+1}}b_{[k+1,\,p]}$$

$$+\sum_{q=k+2}^{l-1}(s_{[q],2}+\max\{0,\, s_{[q],1}+a_{[q,1]}-s_{[q],2}-b_{[q-1,\,\beta_{q-1}]}\}$$

$$+\sum_{p=2}^{\beta_q}\max\{0,\, a_{[q,\,p]}-b_{[q,\,p-1]}\}+\sum_{p=1}^{\beta_q}b_{[q,\,p]})+\max\{0,\, s_{[k],1}+a_{[k,1]}-s_{[k],2}-b_{[l-1,\,\beta_{l-1}]}\}$$

$$-\max\{0,\, s_{[k],1}+a_{[k,1]}-s_{[k],2}-b_{[k-1,\,\beta_{k-1}]}\}+L_i(k),$$

$$L_{\max}(\sigma_2)=\max\{0,\, a_{[k+1,1]}-b_{[k-1,\,\beta_{k-1}]}\}+\max\{0,\, s_{[k],1}+a_{[k,1]}-s_{[k],2}-b_{[l-1,\,\beta_{l-1}]}\}$$

$$+\max\{0,\, a_{[l,1]}-b_{[k,\,\beta_k]}\}-\max\{s_{[k],2},\, s_{[k],1}+a_{[k,1]}-b_{[k-1,\,\beta_{k-1}]}\}$$

$$-\max\{\, s_{[k+1],2},\, s_{[k+1],1}+a_{[k+1,1]}-b_{[k,\,\beta_k]}\}$$

$$-\max\{0,\, s_{[l],1}+a_{[l,1]}-s_{[l],2}-b_{[l-1,\,\beta_{l-1}]}\}+L_{\max}. \qquad (3)$$

3) The batches in the (k–1)st and (k+1)st positions of the schedule are not of the same class, but the batches in the (l–1)st and (l+1)st positions are. Then, L_f, L_b and $L_{max}(\sigma_2)$ are the same as those in 1), but the value of $L_{max}(\sigma_1)$ is changed to

$$L_{\max}(\sigma_1) = \max\{0,\ a_{[l,1]} - b_{[k,\beta_k]}\} + \max\{0,\ s_{[k+1],1} + a_{[k+1,1]} - s_{[k+1],2} - b_{[l,\beta_l]}\}$$

$$+ \max\{0,\ a_{[l+1,1]} - b_{[l-1,\beta_{l-1}]}\} - \max\{s_{[l],2},\ s_{[l],1} + a_{[l,1]} - b_{[l-1,\beta_{l-1}]}\}$$

$$- \max\{0,\ s_{[k+1],1} + a_{[k+1,1]} - s_{[k+1],2} - b_{[k,\beta_k]}\} - \max\{s_{[l+1],2},\ s_{[l+1],1}$$

$$+ a_{[l+1,1]} - b_{[l,\beta_l]}\} + L_{\max}. \qquad (4)$$

4) The batches in the (k–1)st and (k+1)st positions of the schedule are of the same class, and so are the batches in the (l–1)st and (l+1)st positions of the schedule. Then, L_f is the same as that in 1), L_b and $L_{max}(\sigma_2)$ are the same as those in 2), and $L_{max}(\sigma_1)$ is the same as that in 3).

By the above discussion, we analyze the changes of the maximum lateness of the merged positions and that of the whole schedule after two batches of the same class are merged. From (1) to (4), we notice that if neglecting the job processing times, the selection of the batches with the larger class setup time on machine 2 to merge will yield a better schedule.

Property 2 In a BEDD schedule, its jobs satisfy

$$\max\{0,\ a_{[l,1]} - b_{[k,\beta_k]}\} + \max\{0,\ s_{[k+1],1} + a_{[k+1,1]} - s_{[k+1],2} - b_{[l,\beta_l]}\}$$

$$\le \max\{0,\ s_{[k+1],1} + a_{[k+1,1]} - s_{[k+1],2} - b_{[k,\beta_k]}\} + \max\{s_{[l],2},\ a_{[l,1]} + s_{[l],1} - b_{[m-1,\beta_{m-1}]}\}, \quad (5)$$

where $B_i(k)$, $B_i(l)$ and $B_i(m)$ are three batches of class i, $k < (l-1)$ and $l < (m-1)$. For batch $B_i(l)$,

i) If its $L_f < L_b$, then, the maximum lateness of the batches in the (k+1)st to $(m-1)$ st merged positions resulting from a forward merge of $B_i(l)$ is less than that resulting from a backward merge of $B_i(l)$;

ii) If its $L_f > L_b$, then, the maximum lateness of the batches in the (k+1)st to $(m-1)$ st merged positions resulting from a backward merge of $B_i(l)$ is less than that resulting from a forward merge of $B_i(l)$.

Proof i) Let L_1 denote the maximum lateness of the batches in the (k+1)st to $(m-1)$ st positions of a BEDD schedule. Let $C1_x(u)$, where $l+1 \le u \le m-1$, denote the completion time of $B_x(u)$ if $B_i(l)$ is merged forward, $C2_i(l)$ the completion time of batch $B_i(l)$ if $B_i(l)$ is

merged backward. We have

$$C1_i(u) = C_i(k) + \max\{0,\, a_{[l,\,1]} - b_{[k,\,\beta_k]}\} + \sum_{p=2}^{\beta_l} \max\{0,\, a_{[l,\,p]} - b_{[l,\,p-1]}\} + \sum_{p=1}^{\beta_l} b_{[l,\,p]} + s_{[k+1],\,2}$$

$$+ \max\{0,\, s_{[k+1],\,1} + a_{[k+1,\,1]} - s_{[k+1],\,2} - b_{[l,\,\beta_l]}\} + \sum_{p=2}^{\beta_{k+1}} \max\{0,\, a_{[k+1,\,p]} - b_{[k+1,\,p-1]}\}$$

$$+ \sum_{p=1}^{\beta_{k+1}} b_{[k+1,\,p]} + \sum_{q=k+2}^{l-1} (s_{[q],\,2} + \max\{0,\, s_{[q],\,1} + a_{[q,\,1]} - s_{[q],\,2} - b_{[q-1,\,\beta_{q-1}]}\}$$

$$+ \sum_{p=2}^{\beta_q} \max\{0,\, a_{[q,\,p]} - b_{[q,\,p-1]}\} + \sum_{p=1}^{\beta_q} b_{[q,\,p]}) + s_{[l+1],\,2}$$

$$+ \max\{0,\, s_{[l+1],\,1} + a_{[l+1,\,1]} - s_{[l+1],\,2} - b_{[l-1,\,\beta_{l-1}]}\} + \sum_{p=2}^{\beta_{l+1}} \max\{0,\, a_{[l+1,\,p]} - b_{[l+1,\,p-1]}\}$$

$$+ \sum_{p=1}^{\beta_{l+1}} b_{[l+1,\,p]} + \sum_{q=l+2}^{u} (s_{[q],\,2} + \max\{0,\, s_{[q],\,1} + a_{[q,\,1]} - s_{[q],\,2} - b_{[q-1,\,\beta_{q-1}]}\}$$

$$+ \sum_{p=2}^{\beta_q} \max\{0,\, a_{[q,\,p]} - b_{[q,\,p-1]}\} + \sum_{p=1}^{\beta_q} b_{[q,\,p]})\,, \tag{6}$$

$$C2_i(l) = C_i(k) + s_{[k+1],2} + \max\{0,\, s_{[k+1],1} + a_{[k+1,1]} - s_{[k+1],2} - b_{[k,\,\beta_k]}\} + \sum_{p=2}^{\beta_{k+1}} \max\{0,\, a_{[k+1,\,p]} - b_{[k+1,\,p-1]}\}$$

$$+ \sum_{p=1}^{\beta_{k+1}} b_{[k+1,\,p]} + \sum_{q=k+2}^{l-1} (s_{[q],\,2} + \max\{0,\, s_{[q],\,1} + a_{[q,\,1]} - s_{[q],\,2} - b_{[q-1,\,\beta_{q-1}]}\}$$

$$+ \sum_{p=2}^{\beta_q} \max\{0,\, a_{[q,\,p]} - b_{[q,\,p-1]}\} + \sum_{p=1}^{\beta_q} b_{[q,\,p]}) + s_{[l+1],\,2}$$

$$+ \max\{0,\, s_{[l+1],\,1} + a_{[l+1,\,1]} - s_{[l+1],\,2} - b_{[l-1,\,\beta_{l-1}]}\} + \sum_{p=2}^{\beta_{l+1}} \max\{0,\, a_{[l+1,\,p]} - b_{[l+1,\,p-1]}\}$$

$$+ \sum_{p=1}^{\beta_{l+1}} b_{[l+1,\,p]} + \sum_{q=l+2}^{m-1} (s_{[q],\,2} + \max\{0,\, s_{[q],\,1} + a_{[q,\,1]} - s_{[q],\,2} - b_{[q-1,\,\beta_{q-1}]}\}$$

$$+ \sum_{p=2}^{\beta_q} \max\{0,\, a_{[q,\,p]} - b_{[q,\,p-1]}\} + \sum_{p=1}^{\beta_q} b_{[q,\,p]}) + s_{[l],\,2} + \max\{0,\, s_{[l],\,1} + a_{[l,\,1]} - s_{[l],\,2} - b_{[m-1,\,\beta_{m-1}]}\}$$

$$+ \sum_{p=2}^{\beta_l} \max\{0,\, a_{[l,\,p]} - b_{[l,\,p-1]}\} + \sum_{p=1}^{\beta_l} b_{[l,\,p]}\,. \tag{7}$$

From (5), (6) and (7), we have $C1_x(u) < C2_i(l)$ for $u = l+1, \cdots, m-1$. And in a BEDD schedule, $\delta_u \geq \delta_l$ if $u > l$, then,

$$L_2 := \max\{L_x(u) \mid l+1 \leq u \leq m-1, x \neq i\} < L_b .$$

For batch B_i (l), if its $L_f < L_b$, consider the following two cases:

Case 1. L_1 occurs in a batch between the (k+1)st and $(l-1)$ st positions of the BEDD schedule. A forward merge of B_i (l) will cause the maximum lateness of the batches in the (k+1)st to lth merged positions to be equal to L_f; and the maximum lateness in the (l+1)st to $(m-1)$ st batches is L_2. A backward merge of $B_i(l)$ will cause the maximum lateness of batches in the (k+1)st to $(m-1)$ st positions to be $\max\{L_1, L_b\}$. Since $L_f < L_b$ and $L_2 < L_b$, $\max\{L_f, L_2\} < \max\{L_1, L_b\}$.

Case 2. L_1 occurs in a batch between the lth and the $(m-1)$ st positions of the BEDD schedule. A forward merge of B_i (l) will cause the maximum lateness of the batches in the (k+1)st to $(m-1)$st positions to be equal to $\max\{L_f, L_2\}$; and a backward merge of $B_i(l)$ will have the maximum lateness equal to L_b. From $L_f < L_b$ and $L_2 < L_b$, we have $\max\{L_f, L_2\} < L_b$.

ii) Similar to the proof of i). □

In a schedule, for the batches of the same class before the batch with the maximum lateness, we may merge them to reduce setup times. The merging procedure can be performed until the objective cannot be improved any more. Property 2 gives us the preference rules of merging two batches for three batches of the same class.

For the studied problem, sequencing the jobs in each batch in increasing order of the job due dates is not an optimal schedule. So we derive a local dominance rule for sequencing the jobs in each batch. The following theorem will be proved by the technique of swapping two adjacent jobs in a batch.

Theorem 1 If job (k, i) and job (k, j) are adjacent in the kth batch in a schedule, then job (k, i) should precede job (k, j) for minimizing the maximum lateness, if $a_{ki} \leq a_{kj}$, $b_{ki} \leq b_{kj}$, $a_{kj} \leq b_{ki}$,

and $d_{ki} \le d_{kj}$.

Proof We assume that job (k, i) is in an arbitrary position τ and job (k, j) in position $\tau+1$ of the kth batch in schedule σ_1, and schedule σ_2 is exactly the same as σ_1 except that job (k, i) and job (k, j) are interchanged.

Since the two schedules have the same jobs in the first position to the $(\tau-1)$st position of the kth batch, we can set $C_{[k,\tau-1]}(\sigma_1) = C_{[k,\tau-1]}(\sigma_2) = h$.

i) For the case $\tau > 1$.

If the conditions of the theorem hold, then we have

$$L_{[k,\tau]}(\sigma_1) = C_{[k,\tau]}(\sigma_1) - d_{ki} = h + \max\{0, a_{ki} - b_{[k,\tau-1]}\} + b_{ki} - d_{ki}, \tag{8}$$

$$\begin{aligned} L_{[k,\tau+1]}(\sigma_1) &= C_{[k,\tau+1]}(\sigma_1) - d_{kj} = h + \max\{0, a_{ki} - b_{[k,\tau-1]}\} + \max\{0, a_{kj} - b_{ki}\} + b_{ki} + b_{kj} - d_{kj} \\ &= h + \max\{0, a_{ki} - b_{[k,\tau-1]}\} + b_{ki} + b_{kj} - d_{kj}, \end{aligned} \tag{9}$$

$$\begin{aligned} L_{[k,\tau+1]}(\sigma_2) &= C_{[k,\tau+1]}(\sigma_2) - d_{ki} = h + \max\{0, a_{kj} - b_{[k,\tau-1]}\} + \max\{0, a_{ki} - b_{kj}\} + b_{ki} + b_{kj} - d_{ki} \\ &= h + \max\{0, a_{kj} - b_{[k,\tau-1]}\} + b_{ki} + b_{kj} - d_{ki}. \end{aligned} \tag{10}$$

From (8) and (10), we have

$$L_{[k,\tau]}(\sigma_1) \le L_{[k,\tau+1]}(\sigma_2).$$

From (9) and (10), we have

$$L_{[k,\tau+1]}(\sigma_1) \le L_{[k,\tau+1]}(\sigma_2).$$

For $q = \tau+2, \cdots, \beta_k$, we have

$$\begin{aligned} L_{[k,q]}(\sigma_1) &= C_{[k,q]}(\sigma_1) - d_{[k,q]} \\ &= h + \max\{0, a_{ki} - b_{[k,\tau-1]}\} + \max\{0, a_{[k,\tau+2]} - b_{kj}\} + b_{ki} + b_{kj} \\ &\quad + \sum_{p=\tau+3}^{q} \max\{0, a_{[k,p]} - b_{[k,p-1]}\} + \sum_{p=\tau+2}^{q} b_{[k,p]} - d_{[k,q]}, \end{aligned} \tag{11}$$

$$\begin{aligned} L_{[k,q]}(\sigma_2) &= C_{[k,q]}(\sigma_2) - d_{[k,q]} \\ &= h + \max\{0, a_{kj} - b_{[k,\tau-1]}\} + \max\{0, a_{[k,\tau+2]} - b_{ki}\} + b_{ki} + b_{kj} \end{aligned}$$

$$+\sum_{p=\tau+3}^{q}\max\{0,\, a_{[k,\,p]}-b_{[k,\,p-1]}\}+\sum_{p=\tau+2}^{q}b_{[k,\,p]}-d_{[k,\,q]}. \tag{12}$$

Therefore, from (11) and (12), we have

$$L_{[k,\,q]}(\sigma_1)\le L_{[k,q]}(\sigma_2),\ \text{ for }\ q=\tau+2,\cdots,\beta_k.$$

Similarly, for the case that the jobs are sequenced after the *k*th batch, we can also prove

$$L_{[m,\,q]}(\sigma_1)\le L_{[m,q]}(\sigma_2),\ \text{for}\ \ q=1,\ 2,\cdots,\beta_m,\ m=k+1,\cdots,r.$$

ii) For the case $\tau=1$.

Similar to the case i), we can prove the conclusion of the theorem. □

For the two batches of the same class in a schedule, we will derive a dominance rule concerning the last job of the preceding batch and the first job of the following batch. The dominance rule can be proved by moving the jobs' positions, namely the last job of the preceding batch to the first position of the succeeding batch or the first job of the succeeding batch to the last position of the preceding batch.

Theorem 2 Assume that the *i*th and *j*th batches in a schedule belong to the same class and we wish to minimize the maximum lateness.

i) If the following conditions hold:

$$a_{[i,\,\beta_i]}\ge s_{[i+1],\,1}+a_{[i+1,\,1]}-s_{[i+1],\,2}, \tag{13}$$

$$\begin{aligned}&\max\{0,\, s_{[i+1],\,1}+a_{[i+1,\,1]}-s_{[i+1],\,2}-b_{[i,\,\beta_i]}\}+\max\{0,\, s_{[j],\,1}+a_{[j,\,1]}-s_{[j],\,2}-b_{[j-1,\,\beta_{j-1}]}\}\\ &\quad\ge\max\{0,\, s_{[j],\,1}+a_{[i,\,\beta_i]}-s_{[j],\,2}-b_{[j-1,\,\beta_{j-1}]}\}+\max\{0,\, a_{[j,\,1]}-b_{[i,\,\beta_i]}\},\end{aligned} \tag{14}$$

and

$$\begin{aligned}&\max\{0,\, s_{[i+1],\,1}+a_{[i+1,\,1]}-s_{[i+1],\,2}-b_{[i,\,\beta_i]}\}+d_{[i,\,\beta_i]}\\ &\quad\ge\max\{s_{[j],\,2},\, s_{[j],\,1}+a_{[i,\,\beta_i]}-b_{[j-1,\,\beta_{j-1}]}\}+\Delta_{i+1,\,j-1},\end{aligned} \tag{15}$$

then the last job of the *i*th batch should move to the first position of the *j*th batch.

ii) If the following conditions hold:

$$a_{[j,\,1]}\le s_{[i+1],\,1}+a_{[i+1,\,1]}-s_{[i+1],\,2},$$

$$\max\{0,\, s_{[i+1],1} + a_{[i+1,1]} - s_{[i+1],2} - b_{[j,1]}\} + \max\{0,\, s_{[j],1} + a_{[j,2]} - s_{[j],2} - b_{[j-1,\,\beta_{j-1}]}\}$$

$$\le \max\{0,\, s_{[j],1} + a_{[j,1]} - s_{[j],2} - b_{[j-1,\,\beta_{j-1}]}\} + \max\{0,\, a_{[j,2]} - b_{[j,1]}\},$$

and

$$\max\{0,\, s_{[i+1],1} + a_{[i+1,1]} - s_{[i+1],2} - b_{[j,1]}\} + d_{[j,1]}$$

$$\le \max\{s_{[j],2},\, s_{[j],1} + a_{[j,1]} - b_{[j-1,\,\beta_{j-1}]}\} + \Delta_{i+1,\,j-1} \ ,$$

then the first job of the *j*th batch should move to the last position of the *i*th batch.

Proof i) Let the maximum lateness of the (*i*+1)st through (*j*–1)st batches in a schedule be $\psi_{[i+1,\,j-1]}$. We have

$$C_{[j-1]} = C_{[i,\,\beta_i-1]} + \max\{0,\, a_{[i,\,\beta_i]} - b_{[i,\,\beta_i-1]}\} + b_{[i,\,\beta_i]} + s_{[i+1],2} + \max\{0,\, s_{[i+1],1} + a_{[i+1,1]}$$

$$- s_{[i+1],2} - b_{[i,\,\beta_i]}\} + \sum_{q=i+2}^{j-1}(s_{[q],2} + \max\{0,\, s_{[q],1} + a_{[q,1]} - s_{[q],2} - b_{[q-1,\,\beta_{q-1}]}\})$$

$$+ \sum_{q=i+1}^{j-1}\sum_{p=2}^{\beta_q}\max\{0,\, a_{[q,\,p]} - b_{[q,\,p-1]}\} + \sum_{q=i+1}^{j-1}\sum_{p=1}^{\beta_q} b_{[q,\,p]} \ ,$$

$$\psi_{[i+1,\,j-1]} = C_{[j-1]} - \Delta_{i+1,\,j-1}.$$

The maximum lateness of the *j*th batch, $\psi_{[j]}$, is

$$\psi_{[j]} = C_{[j-1]} + s_{[j],2} + \max\{0,\, s_{[j],1} + a_{[j,1]} - s_{[j],2} - b_{[j-1,\,\beta_{j-1}]}\}$$

$$+ \sum_{p=2}^{\beta_j}\max\{0,\, a_{[j,\,p]} - b_{[j,\,p-1]}\} + \sum_{p=1}^{\beta_j} b_{[j,\,p]} - \delta_j \ .$$

Consider the effect of moving job $[i, \beta_i]$ to the first position of the *j*th batch. The maximum lateness of the first batch to the (β_i–1)st job of the *i*th batch will remain unchanged. Let the maximum lateness of the (*i* +1)st through (*j*–1)st batches after the change be $\psi 1_{[i+1,\,j-1]}$. We have

$$C1_{[j-1]} = C_{[i,\,\beta_i-1]} + s_{[i+1],2} + \max\{0,\, s_{[i+1],1} + a_{[i+1,1]} - s_{[i+1],2} - b_{[i,\,\beta_i-1]}\}$$

$$+\sum_{q=i+2}^{j-1}\max\{s_{[q],2},\, s_{[q],1}+a_{[q,1]}-b_{[q-1,\,\beta_{q-1}]}\}+\sum_{q=i+1}^{j-1}\sum_{p=2}^{\beta_q}\max\{0,\, a_{[q,\,p]}-b_{[q,\,p-1]}\}$$

$$+\sum_{q=i+1}^{j-1}\sum_{p=1}^{\beta_q}b_{[q,\,p]},$$

$$\psi 1_{[i+1,\,j-1]}=C1_{[j-1]}-\Delta_{i+1,j-1}.$$

The lateness of job $[i,\beta_i]$ after the change, $L1_{[i,\beta_i]}$, is

$$L1_{[i,\,\beta_i]}=C1_{[j-1]}+s_{[j],2}+\max\{0,\, s_{[j],1}+a_{[i,\,\beta_i]}-s_{[j],2}-b_{[j-1,\,\beta_{j-1}]}\}+b_{[i,\,\beta_i]}-d_{[i,\,\beta_i]}.$$

The maximum lateness of the jth batch after change, $\psi 1_{[j]}$, is

$$\psi 1_{[j]}=C1_{[j-1]}+s_{[j],2}+\max\{0,\, s_{[j],1}+a_{[i,\,\beta_i]}-s_{[j],2}-b_{[j-1,\,\beta_{j-1}]}\}+b_{[i,\,\beta_i]}$$

$$+\max\{0,\, a_{[j,1]}-b_{[i,\,\beta_i]}\}+\sum_{p=2}^{\beta_j}\max\{0,\, a_{[j,\,p]}-b_{[j,\,p-1]}\}+\sum_{p=1}^{\beta_j}b_{[j,\,p]}-\delta_j.$$

It is obvious that

$$\psi 1_{[i+1,\,j-1]}\le\psi_{[i+1,\,j-1]}.$$

From (13) and (15), we have

$$L1_{[i,\,\beta_i]}\le\psi_{[i+1,\,j-1]}.$$

In addition, from (13) and (14), we have

$$\psi 1_{[j]}\le\psi_{[j]}.$$

For the mth batch in the schedule with $m>j$, its maximum lateness will either remain unchanged or decrease after the change.

ii) Similar to i), we can prove the conclusion of the theorem. □

4. Heuristic algorithm

Since our studied scheduling problem is NP-hard, it is very unlikely that efficient algorithms exist for solving it exactly. However, the theoretical results derived in Section 3 are useful for designing heuristic algorithms. Property 2 provides guidelines on the beneficial merging of batches. Theorems 1 and 2 offer hints that the EDD and BEDD rules are important factors for obtaining an optimal solution and provide methods for improving schedules.

4.1 Description of heuristic algorithm

The heuristic algorithm proceeds to solve the problem by first finding an initial schedule and then applying the merging properties and the theorems to improve the initial schedule.

The details of the heuristic procedure are described below.

Step 1. Determine an initial schedule through sequencing jobs in increasing order of the job due dates. Compute the maximum lateness of the schedule, L_{max}, and determine the position m of the batch with the maximum lateness. In case of ties, choose the batch with the least m.

Step 2. Assume that the initial schedule consists of n batches. Take the first m batches to form a partial schedule, $P1$, and the remaining $(n - m)$ batches to form a partial schedule, $P2$.

Step 3. For partial schedule $P1$, select the class with the largest setup time on machine 2, which has not been evaluated for merging yet and go to step 4. If no such batches can be found, go to step 5.

Step 4. Apply Property 1 to compute L_f and L_b of each batch for the selected job class (only L_b for the first batch and only L_f for the last batch); choose the batches with the smallest value of all L_f and L_b, apply Property 2 to merge two batches if the maximum lateness of the schedule after the two batches are merged is less than L_{max}; then sequence all batches such that the schedule becomes a BEDD schedule and update L_{max}, go to Step 4; otherwise, go to Step 3.

Step 5. For each batch i in the schedule, check:

Start at the first job of this batch, for two adjacent jobs $[i, j]$ and $[i, j+1]$, if they satisfy the conditions of Theorem 1, go to the next two adjacent jobs $[i, j+1]$ and $[i, j+2]$. Otherwise, switch their order, and compute the maximum lateness of the schedule, $L'_{\max}$. If $L'_{\max} \geq L_{\max}$, then they are switched back to their old positions; otherwise, let $L_{\max} = L'_{\max}$. Go to next two adjacent jobs $[i, j+1]$ and $[i, j+2]$.

Step 6. Append $P2$ to $P1$. Assume there are N batches in this schedule. For $i = 1$ to $N - 2$ and for $k = i + 2$ to N, check:

If batch k is of the same class as that of batch i, check the conditions i) and ii) in Theorem 2. If i) holds, move job $[i, \beta_i]$ ahead of batch k; if ii) holds, move job $[k, 1]$ to follow

batch i.

Finally, calculate the maximum lateness of the schedule, L_{max}. Stop.

In step 1, $O(n\log n)$ time is required to construct an EDD schedule. In steps 3 and 4, the time complexity is no more than $O(n^2\log n)$. In steps 5 and 6, the computing time is at most $O(n)$. Therefore, the time complexity of the heuristic is $O(n^2\log n)$.

4.2 A numerical example

To illustrate the heuristic algorithm, we give an instance of the considered scheduling problem, which consists of 20 jobs belonging to 4 different job classes. The processing times on machines 1 and 2, the due date of each job, and the setup times on machines 1 and 2 of each class are shown in Table 1. The procedures carried out to solve the instance are detailed below.

Steps 1 and 2. Sequence the jobs in increasing order of the job due dates, which is a BEDD schedule. The 20th job has the maximum lateness of 808. So the partial schedule $P1$ is formed with 18 batches, which are

$B_1(1)$ = {job (1, 3) }, $B_1(7)$ = {job (1, 5)}, $B_1(9)$ = {job (1, 4)},

$B_1(15)$ = {job (1, 2)}, $B_1(17)$ = {job (1, 1)};

$B_2(4)$ = {job (2, 2)}, $B_2(6)$ = {job (2, 4), job (2, 3)},

$B_2(10)$ = {job (2, 1)}, $B_2(14)$ = {job (2, 5)};

$B_3(3)$ = {job (3, 4)}, $B_3(8)$ = {job (3, 5)}, $B_3(12)$ = {job (3, 3)},

$B_3(16)$ = {job (3, 1)}, $B_3(18)$ = {job (3, 2)};

$B_4(2)$ = {job (4, 2), job (4, 5)}, $B_4(5)$ = {job (4, 1) },

$B_4(11)$ = {job (4, 3)}, $B_4(13)$ = {job (4, 4)}.

The due dates of these batches are detailed in Table 2. The partial schedule $P2$ is an empty set in this instance.

Steps 3 and 4. For a partial BEDD schedule, according to the largest setup time on machine 2, select class 3 with setup time 85 on machine 2.

For batches {$B_3(3)$, $B_3(8)$, $B_3(12)$, $B_3(16)$, $B_3(18)$}, apply Property 1 to calculate L_b = 478 for $B_3(3)$, {L_b, L_f}={425, 51} for $B_3(8)$, {L_b, L_f}={591, 195} for $B_3(12)$, {L_b, L_f}={656, 536} for $B_3(16)$, L_f =780 for $B_3(18)$. Apply Property 2 to select $B_3(8)$ for forward merging.

After merging, the maximum lateness of the current schedule is 600. Sequence all batches again to yield a BEDD schedule.

Repeat this procedure, in turn: forward merge $B_3(16)$, obtaining L_{max} = 434; forward merge $B_3(12)$-$B_3(16)$, obtaining L_{max} = 289; and forward merge $B_3(18)$, obtaining L_{max} = 266.

Select class 1: no any merge occurs.

Select class 4: forward merge $B_4(5)$, obtaining L_{max} = 207.

Select class 2: forward merge $B_2(14)$, obtaining L_{max} = 148.

Finally, we obtain the schedule: [$B_1(1)$, $B_4(2)$-$B_4(5)$, $B_2(4)$, $B_3(3)$-$B_3(8)$-$B_3(12)$-$B_3(16)$-$B_3(18)$, $B_2(6)$, $B_1(7)$-$B_1(9)$, $B_2(10)$-$B_2(14)$, $B_4(11)$-$B_4(13)$, $B_1(15)$-$B_1(17)$]. Its maximum lateness is 148.

Step 5. Check the adjacent two jobs for each batch, job (4, 2) and job (4, 5) in $B_4(2)$ require to be exchanged. Denote the changed batch $B_4(2)$ as $\overline{B}_4(2)$ = { job (4, 5), job (4, 2)}. So the current schedule is [$B_1(1)$, $\overline{B}_4(2)$ -$B_4(5)$, $B_2(4)$, $B_3(3)$-$B_3(8)$-$B_3(12)$-$B_3(16)$-$B_3(18)$, $B_2(6)$, $B_1(7)$-$B_1(9)$, $B_2(10)$- $B_2(14)$, $B_4(11)$-$B_4(13)$, $B_1(15)$- $B_1(17)$], whose maximum lateness is 120.

Step 6. Applying Theorem 2 to check the jobs of different batches of the same class, we see that the objective value of the schedule cannot be improved.

Finally, we obtain the schedule: [$B_1(1)$, $\overline{B}_4(2)$ -$B_4(5)$, $B_2(4)$, $B_3(3)$-$B_3(8)$-$B_3(12)$-$B_3(16)$- $B_3(18)$, $B_2(6)$, $B_1(7)$-$B_1(9)$, $B_2(10)$- $B_2(14)$, $B_4(11)$-$B_4(13)$, $B_1(15)$- $B_1(17)$] with a maximum lateness of 120. This is also an optimal schedule.

5. Computational results

Computational experiments were conducted to test the performance of the heuristic approach. The algorithm was coded in VB language and run on a PC Celeron 700.

The processing times on machines 1 and 2 were generated from a uniform integer distribution between 1 and 100. We note that if adding a constant to all the due dates, there will be no difference in the optimal schedule, but the value of L_{max} will change by an amount equal to the constant. In other words, it is essential to pay attention only to the relative distances of the due dates, but not their absolute values in a given problem. Hence, we set the due dates to

be a uniform integer distribution between 1 and $R(\sum a_{ij} + \sum b_{ij})$, where R is a parameter called the factor of the due date range, which indicates the characteristic of the relative distances of the due dates for each other. We chose the R-values in the range between 0.5 and 2.0 in our experiments. Since the size of the setup times may be different from the processing times in practice [6], we generated two integer distributions between 1 and 100, and between 1 and 50, respectively. The latter case is called the problems with reduced setup times.

The algorithm was tested over problem sizes of n = 20, 30 jobs, divided into classes of 2, 4, 5, 10 and 3, 5, 6, 10, respectively. For setting R as 0.5, 1.0, 1.5 and 2.0, and considering the situations with setup times and reduced setup times, 64 cases were examined and 20 replications were randomly generated for each case.

In order to maintain consistency in the computational results of all instances, we anchored the minimum due date of each instance, d_{min}, at zero. Thus, we dealt with the lateness of each job as $L_{ij} = [C_{ij} - (d_{ij} - d_{\min})]$. This also assured that the optimal value of L_{max} is positive. Computational results on the maximal relative deviations and the average relative deviations of the approximate solutions from the optimal solutions were reported, where relative deviation = (approximate maximum lateness – optimal maximum lateness)/optimal maximum lateness. The optimal solutions were obtained by a branch-and-bound algorithm. The branch-and-bound algorithm adopts the depth-first search strategy. In the search procedure, any branch that does not satisfy Theorem 1 or 2, or whose evaluated lower bound is larger than the upper bound is pruned, where the upper bound may be improved by a solution. The average CPU times in seconds also were recorded. Table 3 exhibits the experiment results for setup times between 1 and 100, and Table 4 shows the results for the problems with reduced setup times.

From Table 3, we see that the maximal relative deviations of the approximate solutions from the optimal solutions were within 10%, and the average relative deviations were within 5%. Given the inherent intractability of the studied problem, which involves not only class setup times, but also the constraint of a no-wait processing environment, it is evident that the proposed heuristic algorithm is both efficient and effective.

To further appreciate the experiment results in Table 3, we observe that the maximal

and average relative deviations, and the average CPU times are closely related to the parameter *R*. In the following, we provide some explanations to account for this observation: For the tested problem, we set *R* at 0.5, 1, 1.5 and 2. When the value of *R* is small, the job due dates are closer to one another, rendering the problem more difficult to be solved. Since the procedures used by the heuristic algorithm to search for an optimal solution are carried out according to given rules in advance, the computational cost is small regardless of the problem difficulty, but the relative deviations may be larger in solving harder problems. Because an EDD schedule is closer to the optimal solution for a larger range of the due date distribution, and our heuristic algorithm begins with an EDD schedule, so the larger the value of *R* for a problem, the less is the CPU time required to solve the problem.

For the problems with reduced setup times, the heuristic algorithm exhibited a similar pattern as that for the solving the problems with larger setup times. However, the performance of the heuristic algorithm in this case is a little superior.

6. Conclusions

This paper studied the two-machine flowshop scheduling problem with class setups in a no-wait processing environment to minimize the maximum lateness. After an initial schedule was constructed, two properties were derived to evaluate the effects of forward or backward merge of batches. In addition, two theorems about the dominance relations between jobs were established to improve a schedule. A heuristic utilizing these properties and theorems was proposed. Computational experiments were designed to evaluate the performance of the heuristic. The experimental results reveal that the heuristic approach is very efficient and effective in solving realistic-sized problems.

Acknowledgements

This research was supported in part by The Hong Kong Polytechnic University under a grant from the *Area of Strategic Development in China Business Services*. We are grateful to the constructive comments of an anonymous referee on an earlier version of this paper.

References

[1] N. G. Hall, C. Sriskandarajah, “A survey of machine scheduling problems with blocking and no-wait in process”, *Operations Research*, 1996, 44, 510-525.

[2] R. Logendran, C. Sriskandarajah, “Two-machine group scheduling problem with blocking and anticipatory setups”, *European Journal of Operational Research*, 1993, 69, 467-481.

[3] Y. Sekiguchi, “Optimal schedule in a GT-type flow-shop under series-parallel precedence constraints”, *Journal of the Operation Research Society of Japan*, 1983, 26, 226-251.

[4] D.-L. Yang, M.-S. Chern, “Two-machine flowshop group scheduling problem”, *Computers and Operations Research*, 2000, 27, 975-985.

[5] K. R. Baker, “Scheduling groups of jobs in the two-machine flow-shop”, *Mathematical and Computer Modeling*, 1990, 13, 29-36.

[6] J. C-H. Pan, J.-S. Chen, H.-L. Cheng, “A heuristic approach for single-machine scheduling with due dates and class setups”, *Computers and Operations Research*, 2001, 28, 1111-1130.

[7] S. Webster, K. R. Baker, “Scheduling groups of jobs on a single machine”, *Operations Research*, 1995, 43, 692-704.

[8] C. L. Monma, C. N. Potts, “On the complexity of scheduling with batch setup times”, *Operations Research*, 1989, 37, 798-804.

[9] C. N. Potts and L. N. Van Wassenhove, “Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity”, *Journal of the Operational Research Society*, 1992, 43, 395-406.

[10] C. N. Potts and M. Y. Kovalyov, “Scheduling with batching: a review”, *European Journal of Operational Research*, 2000, 120, 228-249.

Table 1

Data of the example

Class	C_1					C_2				
Job	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)
a_{ij}	55	75	84	71	46	39	31	37	95	67
b_{ij}	57	94	2	44	58	24	35	59	76	5
d_{ij}	1784	1724	51	1368	1119	1376	605	1067	1040	1613
s_{i1}	85					72				
s_{i2}	77					15				

(*Continued*)

Class	C_3					C_4				
Job	(3, 1)	(3, 2)	(3, 3)	(3, 4)	(3, 5)	(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)
a_{ij}	29	18	22	51	20	50	37	12	38	78
b_{ij}	61	87	27	18	62	8	49	63	70	88
d_{ij}	1744	1843	1546	585	1126	862	296	1502	1586	466
s_{i1}	91					41				
s_{i2}	85					60				

Table 2

The initial batches of the example

	$B_1(1)$	$B_4(2)$	$B_3(3)$	$B_2(4)$	$B_4(5)$	$B_2(6)$	$B_1(7)$	$B_3(8)$	$B_1(9)$	$B_2(10)$	$B_4(11)$
δ_{ij}	51	413	585	605	862	1067	1119	1126	1368	1376	1502

(*Continued*)

	$B_3(12)$	$B_4(13)$	$B_2(14)$	$B_1(15)$	$B_3(16)$	$B_1(17)$	$B_3(18)$
δ_{ij}	1546	1586	1613	1724	1744	1784	1843

Table 3

Computational results

No. of jobs	No. of classes	R	L_{max}		
			Max relative deviation (%)	Average relative deviation (%)	Average CPU time (second)
20	2	0.5	6.438	3.362	1.49E-2
		1.0	3.747	0.601	1.07E-2
		1.5	0.000	0.000	4.61E-3
		2.0	0.000	0.000	5.69E-3
	4	0.50	10.23	4.547	1.80E-2
		1.00	8.765	2.732	2.22E-2
		1.50	0.000	0.000	4.22E-3
		2.00	1.829	0.187	2.42E-3
	5	0.50	8.084	2.158	1.41E-2
		1.00	3.681	0.809	9.57E-3
		1.50	2.042	0.566	3.03E-3
		2.00	0.000	0.000	8.04E-3
	10	0.50	6.729	3.196	1.03E-2
		1.00	0.000	0.000	1.07E-3
		1.50	0.000	0.000	4.10E-3
		2.00	1.195	0.120	2.52E-3
30	3	0.50	7.527	2.418	6.13E-2
		1.00	0.563	0.049	3.02E-2
		1.50	0.000	0.000	0.1993
		2.00	1.997	0.286	1.69E-2
	5	0.50	9.760	2.753	7.07E-2
		1.00	0.000	0.000	2.42E-2
		1.50	2.237	0.659	9.83E-3
		2.00	1.588	0.238	1.04E-2
	6	0.50	9.918	4.601	6.41E-2
		1.00	5.402	1.358	2.21E-2
		1.50	0.000	0.000	1.21E-2
		2.00	3.404	0.301	1.04E-2
	10	0.50	12.50	3.355	4.81E-2
		1.00	7.41	1.342	1.64E-2
		1.50	3.180	0.215	5.36E-3
		2.00	0.783	0.067	6.14E-3

Table 4

Computational results with reduced setups

No.of jobs	No.of classes	R	L_{max}		
			Max relative deviation (%)	Average relative deviation (%)	Average CPU time (second)
20	2	0.5	8.812	2.360	9.23E-3
		1.0	3.048	0.493	5.99E-3
		1.5	0.000	0.000	3.91E-3
		2.0	0.000	0.000	5.30E-3
	4	0.50	11.00	3.936	1.94E-2
		1.00	12.46	7.850	9.11E-3
		1.50	6.493	0.896	2.57E-3
		2.00	0.000	0.000	3.21E-3
	5	0.50	6.465	3.305	1.15E-2
		1.00	5.519	1.381	3.20E-3
		1.50	0.000	0.000	2.99E-3
		2.00	0.000	0.000	3.06E-3
	10	0.50	8.001	3.735	1.72E-2
		1.00	1.634	0.285	1.58E-3
		1.50	0.000	0.000	1.82E-3
		2.00	0.656	0.079	2.60E-3
30	3	0.50	4.114	1.907	4.22E-2
		1.00	4.512	0.673	1.57E-2
		1.50	0.000	0.000	1.34E-2
		2.00	0.000	0.000	1.27E-2
	5	0.50	3.365	1.501	4.49E-2
		1.00	2.867	0.573	1.21E-2
		1.50	1.215	0.832	9.67E-3
		2.00	0.609	0.087	7.66E-3
	6	0.50	3.268	0.947	5.71E-3
		1.00	2.419	0.348	9.96E-3
		1.50	0.000	0.000	7.57E-3
		2.00	1.250	0.321	1.03E-2
	10	0.50	8.967	2.901	2.73E-2
		1.00	1.363	0.485	1.18E-2
		1.50	1.170	0.082	4.72E-3
		2.00	0.000	0.000	2.42E-2

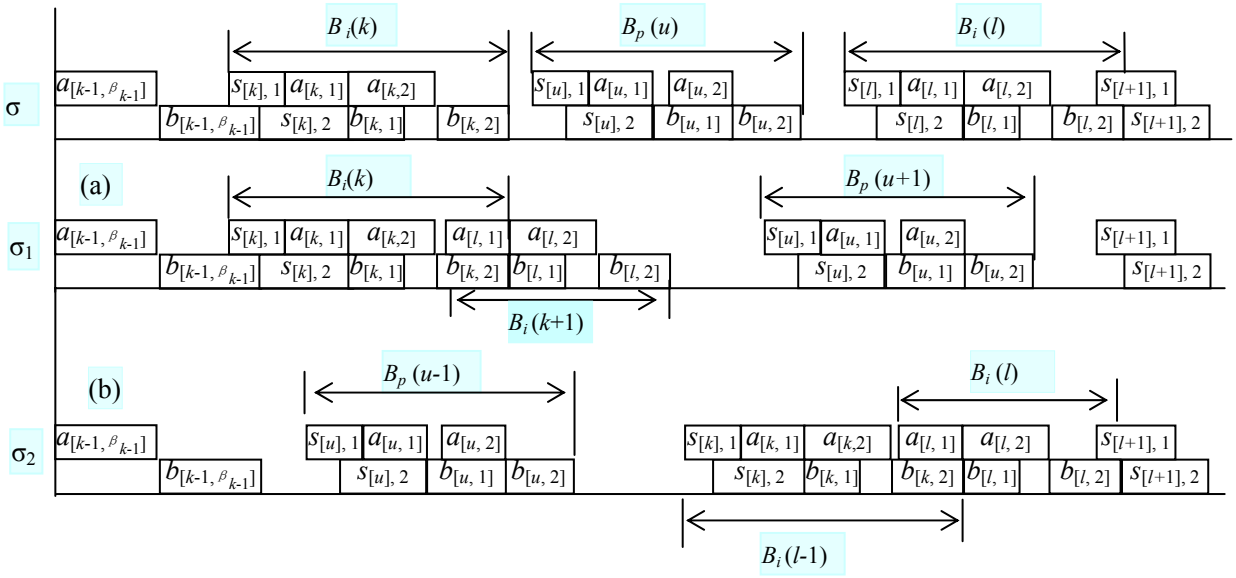

Fig.1. (a) A forward merge of $B_i(l)$, (b) A backward merge of $B_i(k)$.