Two-machine Flowshop Scheduling with Job Class Setups to Minimize Total Flowtime

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 job class setups to minimize the total flowtime. The jobs are classified into classes, and a setup is required on a machine if it switches processing of jobs from one class to another class, but no setup is required if the jobs are from the same class. For some special cases, we derive a number of properties of the optimal solution, based on which we design heuristics and branch-and-bound algorithms to solve these problems. Computational results show that these algorithms are effective in yielding near-optimal or optimal solutions to the tested problems.

Keywords: Flowshop scheduling, job class setups, heuristics, branch-and-bound algorithms, total flowtime.

[#]Corresponding author.

1. Introduction

In many manufacturing settings classes of jobs are processed on one or more machines. On each machine, a setup is required at the beginning of each batch, where a batch is a maximal set of consecutively processed jobs from the same class. A machine can only process one job at a time, and cannot perform any processing while undergoing a setup. For the objective of minimizing the total flowtime, a schedule defines how batches are formed and specifies the processing order of the batches and that of the jobs within the batches.

The single-machine job class scheduling problem to minimize the (weighted) total flowtime has been widely studied by many researchers. For the case of two job classes, Gupta (1984) and Potts (1991) proposed a polynomial time algorithm and a dynamic programming algorithm, respectively. However, Gupta's algorithm is not optimal. For the case of multiple job classes, Gupta (1988), and Ahn and Hyun (1990) proposed different heuristics. Mason and Anderson (1991) developed a branch-and-bound algorithm for the problem to minimize the mean and weighted flowtime. Crauwels et al. (1998), too, proposed a branch-and-bound algorithm, which is superior to that of Mason and Anderson (1991) by using the derived lower bounds from a Lagrangian relaxation of the machine capacity constraints. Crauwels et al. (1997) also developed several local search heuristics, whose performance is superior to the method in Ahn and Hyun (1990). Reviews on this topic have been presented by Potts and Van Wassenhove (1992), Webster and Baker (1995), and Potts and Kovalyov (2000).

In contrast to the existence of many significant research results on the single-machine job class scheduling problem to minimize the (weighted) total flowtime, there have been few attempts to study the problem involving two or more machines (see, for example, Cheng et al., 2000). The two-machine flowshop job class scheduling problem is evidently NP-hard because if all the setup times of the job classes are zero, it becomes the classical two-machine flowshop scheduling problem, which is NP-hard (Gonzalez and Sahni, 1978). For the two-machine flowshop scheduling problem to minimize the mean flowtime, Woo and Yim (1998) proposed an efficient heuristic algorithm. Ho and Gupta (1995) proposed polynomial time algorithms under the dominant machine situation. When each job belongs to a different job class, the problem becomes the two-machine flowshop scheduling problem with sequence-independent

setup times, for which Allahverdi (2000) developed a branch-and-bound algorithm and a heuristic. For the general case, it is highly unlikely that a polynomial algorithm can be found to solve the problem. In this paper we study several special cases of the two-machine flowshop job class scheduling problem and develop efficient algorithms for them. We solve two special cases of the problem and develop several heuristics and branch-and-bound algorithms for the other cases. The efficiency and effectiveness of the heuristics and branch-and-bound algorithms are numerically evaluated.

2. Problem description and notation

For the classical two-machine flowshop scheduling problem with the objective of minimizing the total flowtime, there exists an optimal permutation schedule. To the best of our knowledge, the issue of whether this property can be extended to the corresponding problem with class setups has remained unresolved. However, we assume in this paper that all jobs are available at time zero and no jobs are allowed to pass. In other words, the job sequence is the same on both machines. We are given n jobs that are divided into c classes. Each class i, for $i = 1, 2, \cdots, c$, contains c_i jobs. For the jth job of class i, which we denote by job (i, j), the following notation is defined:

s_{ik} : setup time of class i on machine k, $k = 1, 2$;

a_{ij} : processing time of job (i, j) on machine 1, $i = 1,\ 2,\ \cdots,\ c$, $j = 1,\ 2,\ \cdots,\ c_i$;

b_{ij} : processing time of job (i, j) on machine 2, $i = 1,\ 2,\ \cdots,\ c$, $j = 1,\ 2,\ \cdots,\ c_i$;

C_{ij} : completion time of job (i, j) on machine 2, $i = 1,\ 2,\ \cdots,\ c$, $j = 1,\ 2,\ \cdots,\ c_i$.

A schedule S is an ordered set of the n jobs. It is convenient to regard a schedule as a sequence of batches, where a batch is a maximal consecutive subsequence of the jobs from the same class in S. Let r denote the number of batches and n_i the number of jobs in the ith batch. And let job $[i, j]$ denote the jth job processed in the ith batch in schedule S. We define

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

$a_{[i,\,j]}$: processing time of job $[i, j]$ on machine 1, $i = 1,\ 2,\ \cdots,\ r$, $j = 1,\ 2,\ \cdots,\ n_i$;

$b_{[i,j]}$: processing time of job $[i,j]$ on machine 2, $i = 1, 2, \cdots, r$, $j = 1, 2, \cdots, n_i$;

$C_{[i,j]}$: completion time of job $[i,j]$ on machine 2, $i = 1, 2, \cdots, r$, $j = 1, 2, \cdots, n_i$.

We now define the total flowtime of the jobs under S as

$$F = \sum_{i=1}^{r} \sum_{j=1}^{n_i} C_{[i,j]}.$$

Let A_{ij} denote the sum of the setup times of the batches in positions 1, 2, $\cdots$, i, and the processing times of the jobs in positions from the first job to job $[i,j]$ on machine 1. Then we have

$$A_{ij} = \sum_{k=1}^{i-1} (s_{[k],1} + \sum_{l=1}^{n_k} a_{[k,l]}) + (s_{[i],1} + \sum_{l=1}^{j} a_{[i,l]}),$$

$$C_{[i,j]} = \begin{cases} \max\{A_{i-1,n_{i-1}} + s_{[i],1} + a_{[i,j]}, C_{[i-1,n_{i-1}]} + s_{[i],2}\} + b_{[i,j]} & \text{if } j = 1 \\ \max\{A_{i,j-1} + a_{[i,j]}, C_{[i,j-1]}\} + b_{[i,j]} & \text{if } j > 1 \end{cases}.$$

In order to unify the notation, we set A_{0j}, A_{10}, $C_{[0,j]}$ and $C_{[1,0]}$ to be zero.

Adopting the notation in Brucker (1995), we denote the two-machine flowshop job class scheduling problem to minimize the total flowtime under study as $F2/class/\sum C_{[i,j]}$.

Suppose that all the processing times of the jobs on machines 1 and 2 are equal to a constant t, and each class setup time on machine 1 is no less than that on machine 2. Then this case can be denoted as $F2/s_{i1} \geq s_{i2}, a_{ij} = b_{ij} = t, class/\sum C_{[i,j]}$.

Suppose that the processing times of all jobs on machines 1 and 2 are equal to a constant t, and each job class setup time on machine 2 is no less than the sum of t and the setup time on machine 1. Then this case can be denoted as $F2/s_{i1} + t \leq s_{i2}, a_{ij} = b_{ij} = t, class/\sum C_{[i,j]}$.

For the classical flowshop scheduling problem where no setups or job classes are involved, Ho and Gupta (1995) have studied two special structure flowshops with dominant machines. The cases considered in this paper are described as follows.

If $s_{i1} \geq s_{i2}$ and $\min\{a_{ij}\} \geq \max\{b_{ij}\}$, for $i = 1, 2, \cdots, c$, $j = 1, 2, \cdots, c_i$, we claim that machine 1 dominates machine 2, denoted by $M_1 \succ M_2$. This case can be denoted as $F2/M_1 \succ M_2, class/\sum C_{[i,j]}$.

If $s_{i2} \geq s_{i1} + a_i$, where $a_i = \min\{a_{ij} \mid \text{any job } (i, j) \text{ whose } b_{ij} = \min\{b_{i1}, b_{i2}, \cdots, b_{ic_i}\}\}$, and $\min\{b_{ij}\} \geq \max\{a_{ij}\}$, for $i = 1, 2, \cdots, c$, $j = 1, 2, \cdots, c_i$, we claim that machine 2 dominates machine 1, denoted by $M_2 \succ M_1$. This case can be denoted as $F2/M_2 \succ M_1, class/\sum C_{[i,j]}$.

Suppose that the processing time of any job on machine 1 is equal to that on machine 2, and the setup time of each class on machine 1 is also equal to that on machine 2. Then this case can be denoted as $F2/s_{i1} = s_{i2}, a_{ij} = b_{ij} = t_{ij}, class/\sum C_{[i,j]}$.

For the first two cases, we will see that the restrictions on the general problem lead to these cases to be solved in polynomial time. It is evident that the last three cases are NP-hard even though they are special cases of the general problem. Furthermore, the cases with dominant machines are natural extensions of the classical problem to the processing environment with class setups, and the last case has a different structure characteristic from the single machine class scheduling problem. Therefore, the derivation of theoretical results and the development of effective heuristics and branch-and-bound algorithms for these three special cases are challenging and valuable. Moreover, the findings of this research may shed light on the general problem and provide hints for its solution.

3. Properties of optimal solutions

In this section we study several special cases of the general problem $F2/class/\sum C_{[i,j]}$. Some properties of the optimal solutions are derived in the following.

3.1 $F2/s_{i1} \geq s_{i2}, a_{ij} = b_{ij} = t, class/\sum C_{[i,j]}$.

Theorem 1 For the $F2/s_{i1} \geq s_{i2}, a_{ij} = b_{ij} = t, class/\sum C_{[i,j]}$ problem, there exists an optimal schedule in which each batch consists of all the jobs of a class, and the batches are sequenced in ascending order of $s_{[i],1}/n_i$.

Proof For this case, the completion time of job $[i, j]$ is

$$C_{[i,\,j]} = \sum_{k=1}^{i} s_{[k],1} + (\sum_{k=1}^{i-1} n_k + j + 1)t\,.$$

The total flowtime is

$$\begin{aligned}
F &= \sum_{i=1}^{r}\sum_{j=1}^{n_i} C_{[i,\,j]} \\
&= \sum_{i=1}^{r}\sum_{j=1}^{n_i}[\sum_{k=1}^{i} s_{[k],1} + (\sum_{k=1}^{i-1} n_k + j + 1)t] \\
&= \sum_{i=1}^{r}(n_i \sum_{k=1}^{i} s_{[k],1}) + \sum_{i=1}^{r}\sum_{j=n_1+\cdots+n_{i-1}+1}^{n_1+\cdots+n_{i-1}+n_i} jt + \sum_{i=1}^{r}\sum_{j=1}^{n_i} t \\
&= [n_1 s_{[1],1} + n_2(s_{[1],1} + s_{[2],1}) + \cdots + n_r(s_{[1],1} + \cdots + s_{[r],1})] + \sum_{j=1}^{n} jt + nt \\
&= \sum_{i=1}^{r}(s_{[i],1}\sum_{k=i}^{r} n_k) + (\sum_{j=1}^{n} j + n)t.
\end{aligned}$$

i) For all schedules that consist of r batches, if a schedule satisfies $s_{[1],1}/n_1 \le s_{[2],1}/n_2 \le \cdots \le s_{[r],1}/n_r$, then it has the smallest value of the total flowtime.

Let σ_1 denote a schedule where the batches are sequenced by $s_{[1],1}/n_1 \le s_{[2],1}/n_2 \le \cdots \le s_{[r],1}/n_r$, and σ_2 denote a schedule that is obtained from σ_1 by interchanging the batches only in positions i and $i+k$ $(k \ge 1)$. Denoting the total flowtime of σ_1 and σ_2 as $F(\sigma_1)$ and $F(\sigma_2)$, respectively, we have

$$F(\sigma_1) = nt + t\sum_{j=1}^{n} j + (s_{[1],1}\sum_{j=1}^{r} n_j + \cdots + s_{[r],1}\sum_{j=r}^{r} n_j),$$

$$\begin{aligned}
F(\sigma_2) = nt + t\sum_{j=1}^{n} j + [&s_{[1],1}\sum_{j=1}^{r} n_j + \cdots + s_{[i-1],1}\sum_{j=i-1}^{r} n_j + s_{[i+k],1}\sum_{j=i}^{r} n_j + s_{[i+1],1}(\sum_{j=i+1}^{r} n_j + n_i - n_{i+k}) + \cdots \\
&+ s_{[i+k-1],1}(\sum_{j=i+k-1}^{r} n_j + n_i - n_{i+k}) + s_{[i],1}(\sum_{j=i+k}^{r} n_j + n_i - n_{i+k}) + (s_{[i+k+1],1}\sum_{j=i+k+1}^{r} n_j + \cdots + s_{[r],1}\sum_{j=r}^{r} n_j).
\end{aligned}$$

Then,

$$\begin{aligned}
F(\sigma_2) - F(\sigma_1) &= (s_{[i+k],1} - s_{[i],1})(n_i + \cdots + n_{i+k} + \cdots + n_r) + (n_i - n_{i+k})(s_{[i+1],1} + \cdots + s_{[i+k-1],1}) \\
&\quad + s_{[i],1}(n_i + n_{i+k+1} + \cdots + n_r) - s_{[i+k],1}(n_{i+k} + n_{i+k+1} + \cdots + n_r) \\
&= [n_i(s_{[i+1],1} + \cdots + s_{[i+k-1],1}) - s_{[i],1}(n_{i+1} + \cdots + n_{i+k-1})] \\
&\quad + [s_{[i+k],1}(n_{i+1} + \cdots + n_{i+k-1}) - n_{i+k}(s_{[i+1],1} + \cdots + s_{[i+k-1],1})] + (s_{[i+k],1}n_i - s_{[i],1}n_{i+k}),
\end{aligned}$$

because σ_1 satisfies the following conditions:

$$s_{[1],1}/n_1 \le s_{[2],1}/n_2 \le \cdots \le s_{[r],1}/n_r,$$

we have

$$\begin{aligned}
&n_i(s_{[i+1],1}+\cdots+s_{[i+k-1],1}) \ge s_{[i],1}(n_{i+1}+\cdots+n_{i+k-1}),\\
&s_{[i+k],1}(n_{i+1}+\cdots+n_{i+k-1}) \ge n_{i+k}(s_{[i+1],1}+\cdots+s_{[i+k-1],1}),\\
&s_{[i+k],1}n_i \ge s_{[i],1}n_{i+k}.
\end{aligned}$$

Therefore, $F(\sigma_2)-F(\sigma_1)\ge 0$.

ii) Given a schedule (denoted as σ_1) in which the batches are sequenced in ascending order of $s_{[i],1}/n_i$, if the kth and lth batches ($k+1<l$) belong to the same job class, then we merge the two batches and again sequence all batches in ascending order of $s_{[i],1}/n_i$. The total flowtime of the resulting schedule (denoted as σ_2) is less than that of the original schedule σ_1.

We assume that the merged batch is in the ith position of schedule σ_2. Obviously, $i \le k$ because $s_{[k],1}/(n_k+n_l) < s_{[k],1}/n_k$. We have

$$F(\sigma_1) = nt + t\sum_{j=1}^{n} j + (s_{[1],1}\sum_{j=1}^{r} n_j + \cdots + s_{[r],1}\sum_{j=r}^{r} n_j)$$

and

$$\begin{aligned}
F(\sigma_2) = nt + t\sum_{j=1}^{n} j &+ s_{[1],1}\sum_{j=1}^{r} n_j + \cdots + s_{[i-1],1}\sum_{j=i-1}^{r} n_j + s_{[k],1}\sum_{j=i}^{r} n_j + s_{[i],1}(\sum_{j=i}^{r} n_j - n_k - n_l)\\
&+ \cdots + s_{[k-1],1}(\sum_{j=k-1}^{r} n_j - n_k - n_l) + s_{[k+1],1}(\sum_{j=k+1}^{r} n_j - n_l) + \cdots + s_{[l-1],1}(\sum_{j=l-1}^{r} n_j - n_l)\\
&+ s_{[l+1],1}\sum_{j=l+1}^{r} n_j + \cdots + s_{[r],1}\sum_{j=r}^{r} n_j.
\end{aligned}$$

Then,

$$\begin{aligned}
F(\sigma_1)-F(\sigma_2) = {} & s_{[l],1}(n_l+\cdots+n_r) + (s_{[i],1}+\cdots+s_{[k-1],1})(n_k+n_l) + (s_{[k+1],1}+\cdots+s_{[l-1],1})n_l\\
& - s_{[k],1}(n_i+n_{i+1}+\cdots+n_{k-1}).
\end{aligned}$$

Note that schedule σ_2 satisfies the following condition:

$$s_{[k],1}/(n_k+n_l) \le s_{[i],1}/n_i \le s_{[i+1],1}/n_{i+1} \le \cdots \le s_{[k-1],1}/n_{k-1}.$$

So, we have

$$s_{[k],1}(n_i+n_{i+1}+\cdots+n_{k-1}) \le (s_{[i],1}+s_{[i+1],1}+\cdots+s_{[k-1],1})(n_k+n_l).$$

Therefore, $F(\sigma_1) - F(\sigma_2) > 0$.

From i) and ii), we reach the conclusion of the theorem. □

According to Theorem 1, an algorithm that treats a job class as a single batch and sequences the batches in ascending order of $s_{[i],1} / n_i$ produces an optimal solution.

3.2 $F2 / s_{i1} + t \le s_{i2}, a_{ij} = b_{ij} = t, class / \sum C_{[i,j]}$.

Theorem 2 For the $F2 / s_{i1} + t \le s_{i2}, a_{ij} = b_{ij} = t, class / \sum C_{[i,j]}$ problem, an optimal schedule can be obtained if a batch consists of all the jobs of a class, and the batches are sequenced in ascending order of $s_{[i],2} / n_i$.

Proof Similar to the proof of Theorem 1. □

According to Theorem 2, an algorithm that treats a job class as a single batch and sequences all batches in ascending order of $s_{[i],2} / n_i$ produces an optimal solution.

3.3 $F2 / M_1 \succ M_2, class / \sum C_{[i,j]}$.

Theorem 3 For the $F2 / M_1 \succ M_2, class / \sum C_{[i,j]}$ problem, there exists an optimal schedule where the jobs in a batch *i* are sequenced in ascending order of $a_{[i,j]}$.

Proof Let σ_1 denote a schedule that comprises *r* batches. Let σ_2 denote a schedule obtained by interchanging only the jobs $[i, j]$ and $[i, j+k]$ $(k \ge 1)$ in schedule σ_1. Obviously, the completion times of the jobs up to job $[i, j-1]$ or after job $[i, j+k]$ in the same positions of σ_1 and σ_2 are equal. In schedule σ_1, we have

$$C(\sigma_1)_{[i,j]} = \sum_{p=1}^{i} s_{[p],1} + \sum_{p=1}^{i-1} \sum_{q=1}^{n_p} a_{[p,q]} + \sum_{q=1}^{j} a_{[i,q]} + b_{[i,j]},$$

$$C(\sigma_1)_{[i,\,j+h]} = \sum_{p=1}^{i} s_{[p],1} + \sum_{p=1}^{i-1}\sum_{q=1}^{n_p} a_{[p,\,q]} + \sum_{q=1}^{j+h} a_{[i,\,q]} + b_{[i,\,j+h]}, \quad \text{for } h = 1, 2, \cdots, k.$$

In schedule σ_2, we have

$$C(\sigma_2)_{[i,\,j]} = \sum_{p=1}^{i} s_{[p],1} + \sum_{p=1}^{i-1}\sum_{q=1}^{n_p} a_{[p,\,q]} + \sum_{q=1}^{j-1} a_{[i,\,q]} + a_{[i,\,j+k]} + b_{[i,\,j+k]},$$

$$C(\sigma_2)_{[i,\,j+h]} = \sum_{p=1}^{i} s_{[p],1} + \sum_{p=1}^{i-1}\sum_{q=1}^{n_p} a_{[p,\,q]} + \sum_{q=1}^{j-1} a_{[i,\,q]} + a_{[i,\,j+k]} + \sum_{q=j+1}^{j+h} a_{[i,\,q]} + b_{[i,\,j+h]}, \text{ for } h = 1, 2, \cdots, k-1,$$

$$C(\sigma_2)_{[i,\,j+k]} = \sum_{p=1}^{i} s_{[p],1} + \sum_{p=1}^{i-1}\sum_{q=1}^{n_p} a_{[p,\,q]} + \sum_{q=1}^{j+k} a_{[i,q]} + b_{[i,\,j]}.$$

Then

$$\sum_{q=j}^{j+k} C(\sigma_2)_{[i,\,q]} = \sum_{q=j}^{i+k} C(\sigma_1)_{[i,\,q]} + k(a_{[i,\,j+k]} - a_{[i,\,j]}).$$

If $a_{[i,j]} \le a_{[i,j+k]}$, then the total flowtime $F(\sigma_1)$ and $F(\sigma_2)$ of schedules σ_1 and σ_2, respectively, have the following relation

$$F(\sigma_1) = \sum_{p=1}^{r}\sum_{q=1}^{n_p} C(\sigma_1)_{[p,\,q]} \le F(\sigma_2) = \sum_{p=1}^{r}\sum_{q=1}^{n_p} C(\sigma_2)_{[p,\,q]}\ .$$

Hence, schedule σ_1 is no worse than schedule σ_2. □

Theorem 4 For the $F2/M_1 \succ M_2, class/\sum C_{[i,\,j]}$ problem, there exists an optimal schedule where the batches are sequenced in ascending order of $(s_{[i],1} + \sum_{j=1}^{n_i} a_{[i,j]})/n_i$.

Proof In the *i*th batch, we have

$$C_{[i,\,j]} = \sum_{l=1}^{i} s_{[l],1} + \sum_{k=1}^{i-1}\sum_{l=1}^{n_k} a_{[k,\,l]} + \sum_{l=1}^{j} a_{[i,\,l]} + b_{[i,\,j]}, \quad \text{for } j = 1, 2, \cdots, n_i,$$

$$\sum_{j=1}^{n_i} C_{[i,\,j]} = n_i \sum_{l=1}^{i} s_{[l],1} + n_i \sum_{k=1}^{i-1}\sum_{l=1}^{n_k} a_{[k,\,l]} + \sum_{l=1}^{n_i} (n_i - l + 1) a_{[i,\,l]} + \sum_{l=1}^{n_i} b_{[i,\,l]}.$$

In the (*i*+1)st batch, we have

$$\sum_{j=1}^{n_{i+1}} C_{[i+1,\,j]} = n_{i+1}\sum_{l=1}^{i+1} s_{[l],1} + n_{i+1}\sum_{k=1}^{i}\sum_{l=1}^{n_k} a_{[k,\,l]} + \sum_{l=1}^{n_{i+1}} (n_{i+1} - l + 1)a_{[i+1,\,l]} + \sum_{l=1}^{n_{i+1}} b_{[i+1,\,l]}\,.$$

Then

$$\sum_{j=1}^{n_i} C_{[i,\,j]} + \sum_{j=1}^{n_{i+1}} C_{[i+1,\,j]} = (n_i + n_{i+1})\sum_{l=1}^{i-1} s_{[l],1} + (n_i + n_{i+1})\sum_{k=1}^{i-1}\sum_{l=1}^{n_k} a_{[k,\,l]} + (n_i + n_{i+1})s_{[i],1} + n_{i+1}s_{[i+1],1}$$
$$+ n_{i+1}\sum_{l=1}^{n_i} a_{[i,\,l]} + \sum_{l=1}^{n_i} (n_i - l + 1)a_{[i,\,l]} + \sum_{l=1}^{n_{i+1}} (n_{i+1} - l + 1)a_{[i+1,\,l]} + \sum_{l=1}^{n_i} b_{[i,\,l]} + \sum_{l=1}^{n_{i+1}} b_{[i+1,\,l]}.$$

Interchanging the *i*th and (*i*+1)st batches, the sum of the job completion times within the two batches is

$$\sum_{j=1}^{n_{i+1}} C'_{[i,\,j]} + \sum_{j=1}^{n_i} C'_{[i+1,j]} = (n_i + n_{i+1})\sum_{l=1}^{i-1} s_{[l],1} + (n_i + n_{i+1})\sum_{k=1}^{i-1}\sum_{l=1}^{n_k} a_{[k,\,l]} + (n_i + n_{i+1})s_{[i+1],1} + n_i s_{[i],1}$$
$$+ n_i\sum_{j=1}^{n_{i+1}} a_{[i+1,\,j]} + \sum_{l=1}^{n_{i+1}} (n_{i+1} - l + 1)a_{[i+1,\,l]} + \sum_{l=1}^{n_i} (n_i - l + 1)a_{[i,\,l]} + \sum_{l=1}^{n_{i+1}} b_{[i+1,\,l]} + \sum_{l=1}^{n_i} b_{[i,\,l]}.$$

Clearly, the completion times of the jobs before the *i*th batch or after the (*i*+1)st batch are not changed after interchanging the *i*th and (*i*+1)st batches. Then, for the total completion time F and F' before and after interchanging, respectively, we have

$$F - F' = (\sum_{j=1}^{n_i} C_{[i,\,j]} + \sum_{j=1}^{n_{i+1}} C_{[i+1,\,j]}) - (\sum_{j=1}^{n_{i+1}} C'_{[i,\,j]} + \sum_{j=1}^{n_i} C'_{[i+1,\,j]})$$
$$= n_{i+1}(s_{[i],1} + \sum_{j=1}^{n_i} a_{[i,\,j]}) - n_i(s_{[i+1],1} + \sum_{j=1}^{n_{i+1}} a_{[i+1,\,j]}).$$

If $(s_{[i],1} + \sum_{j=1}^{n_i} a_{[i,\,j]})/n_i \le (s_{[i+1],1} + \sum_{j=1}^{n_{i+1}} a_{[i+1,\,j]})/n_{i+1}$, we have $F - F' \le 0$. □

Theorem 5 For the $F2/M_1 \succ M_2, class/\sum C_{[i,\,j]}$ problem, where the jobs in the *i*th and (*i*+*k*)th batches belong to the same job class, there exists an optimal schedule where $a_{[i,n_i]}$ and $a_{[i+k,1]}$ satisfy the following condition

$$a_{[i,\,n_i]} \le (\sum_{l=1}^{k} s_{[i+l],1} + \sum_{l=1}^{k-1}\sum_{j=1}^{n_{i+l}} a_{[i+l,\,j]})/\sum_{l=1}^{k-1} n_{i+l} \le a_{[i+k,1]}. \qquad (1)$$

Proof. In a schedule *S*, we denote all the processed jobs between job [*i, n*$_i$] and job [*i*+*k*, 1] as a partial sequence σ. Let S' be the sequence formed by moving job [*i*+*k*, 1] to the last of the *i*th

batch in S. Clearly, the completion times of the jobs sequenced before job $[i, n_i]$ or after job $[i+k, 1]$ are not changed. Let F and F' denote the total flowtime of S and S', respectively. In schedule S, we have

$$C_{[i+l, j]} = \sum_{h=1}^{i+l} s_{[h],1} + \sum_{h=1}^{i+l-1}\sum_{g=1}^{n_h} a_{[h, g]} + \sum_{g=1}^{j} a_{[i+l, g]} + b_{[i+l, j]}, \quad \text{for} \quad l = 1, 2, \cdots, k-1,\ j = 1, 2, \cdots, n_{i+l},$$

$$C_{[i+k, 1]} = \sum_{h=1}^{i+k} s_{[h],1} + \sum_{h=1}^{i+k-1}\sum_{g=1}^{n_h} a_{[h, g]} + a_{[i+k, 1]} + b_{[i+k, 1]}.$$

In schedule S', we have

$$\begin{aligned} C'_{[i, n_i+1]} &= \sum_{h=1}^{i} s_{[h],1} + \sum_{h=1}^{i}\sum_{j=1}^{n_h} a_{[h, j]} + a_{[i+k, 1]} + b_{[i+k, 1]} \\ &= C_{[i+k, 1]} - (\sum_{h=1}^{k-1}\sum_{j=1}^{n_{i+h}} a_{[i+h, j]} + \sum_{h=1}^{k} s_{[i+h],1}), \\ C'_{[i+l, j]} &= C_{[i+l, j]} + a_{[i+k, 1]}, \ \text{for} \ l = 1, 2, \cdots, k-1,\ j = 1, 2, \cdots, n_{i+l}. \end{aligned}$$

Then,

$$\sum_{l=1}^{k-1}\sum_{j=1}^{n_{i+l}} C'_{[i+l, j]} + C'_{[i, n_i+1]} = \sum_{l=1}^{k-1}\sum_{j=1}^{n_{i+l}} C_{[i+l, j]} + a_{[i+k, 1]}\sum_{l=1}^{k-1} n_{i+l} + C_{[i+k, 1]} - (\sum_{h=1}^{k} s_{[i+h],1} + \sum_{h=1}^{k-1}\sum_{j=1}^{n_{i+h}} a_{[i+h, j]}).$$

Therefore,

$$F' - F = a_{[i+k, 1]}\sum_{l=1}^{k-1} n_{i+l} - (\sum_{h=1}^{k} s_{[i+h],1} + \sum_{h=1}^{k-1}\sum_{j=1}^{n_{i+h}} a_{[i+h, j]}).$$

Therefore, if the right hand side of inequality (1) holds, i.e., $F' - F \geq 0$, which means that job $[i+k, 1]$ should not be moved to the last position of the ith batch to minimize the total flowtime.

Similar to the above, we can also prove that if the left hand side of inequality (1) holds, job $[i, n_i]$ should not be moved to the beginning position of the $(i+k)$th batch to minimize the total flowtime. □

3.4 $F2/M_2 \succ M_1, class/\sum C_{[i, j]}$.

Theorem 6 For the $F2/M_2 \succ M_1, class/\sum C_{[i, j]}$ problem, there exists an optimal schedule where the jobs in a batch i are sequenced in ascending order of $b_{[i, j]}$.

Proof Similar to the proof of Theorem 3. □

Theorem 7 For the $F2/M_2 \succ M_1, class/\sum C_{[i,j]}$ problem, there exists an optimal schedule where the batches are sequenced in ascending order of $(s_{[i],2} + \sum_{j=1}^{n_i} b_{[i,j]})/n_i$.

Proof Similar to the proof of Theorem 4. □

Theorem 8 For the $F2/M_2 \succ M_1, class/\sum C_{[i,j]}$ problem, where the jobs of the *i*th and (*i*+*k*)th batches belong to the same job class, there exists an optimal schedule where $b_{[i,n_i]}$ and $b_{[i+k,1]}$ satisfy the following condition

$$b_{[i,n_i]} \le (\sum_{l=1}^{k} s_{[i+l],2} + \sum_{l=1}^{k-1}\sum_{j=1}^{n_{i+l}} b_{[i+l,j]})/\sum_{l=1}^{k-1} n_{i+l} \le b_{[i+k,1]} .$$

Proof Similar to the proof of Theorem 5. □

3.5 $F2/s_{i1} = s_{i2}, a_{ij} = b_{ij} = t_{ij}, class/\sum C_{[i,j]}$.

For convenience, we set $s_{i1} = s_{i2} = s_i$, for $i = 1, 2, \cdots, c$, and $a_{ij} = b_{ij} = t_{ij}$, for $i = 1, 2, \cdots, c$, $j = 1, 2, \cdots, c_i$. For this special case, we can derive some optimal properties in the following.

Theorem 9 For the $F2/s_{i1} = s_{i2}, a_{ij} = b_{ij} = t_{ij}, class/\sum C_{[i,j]}$ problem, there exists an optimal schedule where the jobs in a batch *i* are sequenced in ascending order of $t_{[i,j]}$.

Proof Let σ_1 denote a sequence where the jobs within a batch are sequenced in ascending order of $t_{[i,j]}$. Let σ_2 denote a sequence that is obtained from σ_1 by swapping the jobs [*i*, *j*] and [*i*, *j*+1]. Obviously, the completion times of all jobs before job [*i*, *j*] are equal in σ_1 and σ_2. In schedule σ_1, we have

$$C(\sigma_1)_{[i,j]} = \max\{A_{i,j-1} + t_{[i,j]}, C_{[i,j-1]}\} + t_{[i,j]},$$

$$\begin{aligned} C(\sigma_1)_{[i,j+1]} &= \max\{A_{i,j-1} + t_{[i,j]} + t_{[i,j+1]}, C(\sigma_1)_{[i,j]}\} + t_{[i,j+1]} \\ &= \max\{A_{i,j-1} + t_{[i,j+1]}, A_{i,j-1} + t_{[i,j]}, C_{[i,j-1]}\} + t_{[i,j]} + t_{[i,j+1]} \\ &= \max\{A_{i,j-1} + t_{[i,j+1]}, C_{[i,j-1]}\} + t_{[i,j]} + t_{[i,j+1]}, \end{aligned} \tag{2}$$

$$\begin{aligned} C(\sigma_1)_{[i,j]} + C(\sigma_1)_{[i,j+1]} &= \max\{A_{i,j-1} + t_{[i,j]}, C_{[i,j-1]}\} \\ &\quad + \max\{A_{i,j-1} + t_{[i,j+1]}, C_{[i,j-1]}\} + 2t_{[i,j]} + t_{[i,j+1]}. \end{aligned} \tag{3}$$

In schedule σ_2, we have

$$C(\sigma_2)_{[i,j]} = \max\{A_{i,j-1} + t_{[i,j+1]}, C_{[i,j-1]}\} + t_{[i,j+1]},$$

$$\begin{aligned} C(\sigma_2)_{[i,j+1]} &= \max\{A_{i,j-1} + t_{[i,j+1]} + t_{[i,j]}, C(\sigma_2)_{[i,j]}\} + t_{[i,j]} \\ &= \max\{A_{i,j-1} + t_{[i,j]}, A_{i,j-1} + t_{[i,j+1]}, C_{[i,j-1]}\} + t_{[i,j]} + t_{[i,j+1]} \\ &= \max\{A_{i,j-1} + t_{[i,j+1]}, C_{[i,j-1]}\} + t_{[i,j]} + t_{[i,j+1]}, \end{aligned} \tag{4}$$

$$C(\sigma_2)_{[i,j]} + C(\sigma_2)_{[i,j+1]} = 2\max\{A_{i,j-1} + t_{[i,j+1]}, C_{[i,j-1]}\} + t_{[i,j]} + 2t_{[i,j+1]}. \tag{5}$$

From (2) and (4), we have $C(\sigma_1)_{[i,j+1]} = C(\sigma_2)_{[i,j+1]}$. And, obviously, swapping job $[i,j]$ and job $[i,j+1]$ will not change the completion time of the last of these two jobs on machine 1. So the completion times of all jobs after job $[i,j+1]$ in σ_1 are not changed in σ_2. From (3) and (5), we have

$$C(\sigma_1)_{[i,j]} + C(\sigma_1)_{[i,j+1]} \le C(\sigma_2)_{[i,j]} + C(\sigma_2)_{[i,j+1]}.$$

Then, the total flowtime of σ_1 is no greater than the total flowtime of σ_2. □

Theorem 10 For the $F2/s_{i1} = s_{i2}, a_{ij} = b_{ij} = t_{ij}, class/\sum C_{[i,j]}$ problem, in a sequence where batches i and k are adjacent, let the jobs within batch i (or k) be sequenced in ascending order of their processing times on machine 1 or 2. In order to minimize the total flowtime, batch i should precede batch k, if

$$t_{in_i} \le t_{k1} \quad \text{and} \quad (s_i + \sum_{j=1}^{n_i} t_{ij})/n_i \le (s_k + \sum_{j=1}^{n_k} t_{kj})/n_k. \tag{6}$$

Proof We assume that batches i and k are in positions h and h+1 in a sequence σ_1, respectively, and sequence σ_2 is obtained by swapping batches i and k. Obviously, the completion times of any job before the hth batch are equal in σ_1 and σ_2. For the hth and (h+1)st batches in sequence σ_1, we have

$$C(\sigma_1)_{[h,j]} = \max\{A_{h-1,n_{h-1}} + s_i + t_{i1} + \cdots + t_{ij},\ C_{[h-1,n_{h-1}]} + s_i + t_{i1} + \cdots + t_{i,j-1}\} + t_{ij}$$

$$= (s_i + \sum_{l=1}^{j} t_{il}) + \max\{A_{h-1,\,n_{h-1}} + t_{ij},\ C_{[h-1,\,n_{h-1}]}\},\quad \text{for}\quad j = 1, 2, \cdots, n_i,$$

$$C(\sigma_1)_{[h+1,j]} = \max\{A_{h,n_h} + s_k + t_{k1} + \cdots + t_{kj},\ C(\sigma_1)_{[h,n_i]} + s_k + t_{k1} + \cdots + t_{k,j-1}\} + t_{kj}$$

$$= (s_k + \sum_{l=1}^{j} t_{kl}) + \max\{A_{h,n_k} + t_{kj},\ C(\sigma_1)_{[h,n_i]}\}$$

$$= (s_k + \sum_{l=1}^{j} t_{kl}) + \max\{A_{h-1,n_{h-1}} + (s_i + \sum_{l=1}^{n_i} t_{il}) + t_{kj},\ A_{h-1,n_{h-1}} + (s_i + \sum_{l=1}^{n_i} t_{il}) + t_{in_i},$$

$$C_{[h-1,n_{h-1}]} + (s_i + \sum_{l=1}^{n_i} t_{il})\}$$

$$= (s_i + \sum_{l=1}^{n_i} t_{il}) + (s_k + \sum_{l=1}^{j} t_{kl}) + \max\{A_{h-1,\,n_{h-1}} + t_{kj},\ A_{h-1,n_{h-1}} + t_{in_i},\ C_{[h-1,\,n_{h-1}]}\},$$

$$\text{for } j = 1, 2, \cdots, n_k, \quad (7)$$

$$\sum_{j=1}^{n_i} C(\sigma_1)_{[h,j]} + \sum_{j=1}^{n_k} C(\sigma_1)_{[h+1,j]} = (n_i + n_k)s_i + n_k s_k + n_k \sum_{l=1}^{n_i} t_{il} + \sum_{l=1}^{n_i} (n_i - l + 1)t_{il}$$

$$+ \sum_{l=1}^{n_k} (n_k - l + 1)t_{kl} + \sum_{l=1}^{n_i} \max\{A_{h-1,\,n_{h-1}} + t_{il},\ C_{[h-1,\,n_{h-1}]}\}$$

$$+ \sum_{l=1}^{n_k} \max\{A_{h-1,\,n_{h-1}} + t_{kl},\ A_{h-1,n_{h-1}} + t_{in_i},\ C_{[h-1,\,n_{h-1}]}\}. \quad (8)$$

For the hth and (h+1)st batches in sequence σ_2, similar to the above, we can derive

$$C(\sigma_2)_{[h,j]} = (s_k + \sum_{l=1}^{j} t_{kl}) + \max\{A_{h-1,\,n_{h-1}} + t_{kj},\ C_{[h-1,\,n_{h-1}]}\},\quad \text{for}\quad j = 1, 2, \cdots, n_k,$$

$$C(\sigma_2)_{[h+1,j]} = (s_k + \sum_{l=1}^{n_k} t_{kl}) + (s_i + \sum_{l=1}^{j} t_{il}) + \max\{A_{h-1,\,n_{h-1}} + t_{ij},\ A_{h-1,n_{h-1}} + t_{kn_k},\ C_{[h-1,\,n_{h-1}]}\}.$$

$$\text{for}\quad j = 1, 2, \cdots, n_i, \quad (9)$$

$$\sum_{j=1}^{n_k} C(\sigma_2)_{[h,j]} + \sum_{j=1}^{n_i} C(\sigma_2)_{[h+1,j]} = (n_i + n_k)s_k + n_i s_i + n_i \sum_{l=1}^{n_k} t_{kl} + \sum_{l=1}^{n_k} (n_k - l + 1)t_{kl}$$
$$+ \sum_{l=1}^{n_i} (n_i - l + 1)t_{il} + \sum_{l=1}^{n_k} \max\{A_{h-1,n_{h-1}} + t_{kl}, C_{[h-1,n_{h-1}]}\}$$
$$+ \sum_{l=1}^{n_i} \max\{A_{h-1,n_{h-1}} + t_{il}, A_{h-1,n_{h-1}} + t_{kn_k}, C_{[h-1,n_{h-1}]}\}. \quad (10)$$

From (7) and (9), we have

$$C(\sigma_1)_{[h+1,n_k]} = C(\sigma_2)_{[h+1,n_i]}.$$

Then, the completion times of any job after the (h+1)st batch in σ_1 and σ_2 are equal.

From (6), (8) and (10), we have

$$\sum_{j=1}^{n_i} C(\sigma_1)_{[h,j]} + \sum_{j=1}^{n_k} C(\sigma_1)_{[h+1,j]} \le \sum_{j=1}^{n_i} C(\sigma_2)_{[h+1,j]} + \sum_{j=1}^{n_k} C(\sigma_2)_{[h,j]}.$$

Thus, if the condition of the theorem holds, then the conclusion is valid. □

Theorem 11 For the $F2/s_{i1} = s_{i2}, a_{ij} = b_{ij} = t_{ij}, class/\sum C_{[i,j]}$ problem, jobs within any batches are sequenced in ascending order of their processing times on machine 1 or 2, and the jobs in the ith and (i+k)th batches belong to the same job class. In order to minimize the total flowtime,

i) job [i+k, 1] should move to the last position of the ith batch, if $t_{[i+k,1]} \le t_{[i+l,1]}$, for $l = 1, 2, \cdots, k-1$;

ii) job $[i, n_i]$ should move to the first position of the (i+k)th batch, if $t_{[i,n_i]} \ge t_{[i+l,n_{i+l}]}$, for $l = 1, 2, \cdots, k-1$.

Proof i) Let σ_1 denote the original schedule, and σ_2 a schedule obtained from σ_1 by moving the first job in the (i+k)th batch to the last position of the ith batch. Obviously, the completion times of all the jobs from the first job up to job [i, n_i] in σ_1 and σ_2 are equal. Under the assumptions of the theorem, in schedule σ_1, we have

$$C(\sigma_1)_{[i+1,\,j]} = s_{[i+1]} + \sum_{l=1}^{j} t_{[i+1,\,l]} + \max\{A_{i\,n_i} + t_{[i+1,\,j]},\, C_{[i,\,n_i]}\},\quad \text{for}\ \ j = 1, 2, \cdots, n_{i+1},$$

$$C(\sigma_1)_{[i+l,\,j]} = \max\{A_{i\,n_i} + t_{[i+1,\,n_{i+1}]}, \cdots, A_{i\,n_i} + t_{[i+l-1,\,n_{i+l-1}]}, A_{i\,n_i} + t_{[i+l,\,j]}, C_{[i,\,n_i]}\}$$

$$+ \sum_{h=1}^{l} s_{[i+h]} + \sum_{h=1}^{l-1}\sum_{j=1}^{n_{i+h}} t_{[i+h,\,j]} + \sum_{h=1}^{j} t_{[i+l,\,h]}\ ,\quad \text{for}\ \ l = 2, \cdots, k,\ j = 1, 2, \cdots, n_{i+l}. \quad (11)$$

In schedule σ_2, we have

$$C(\sigma_2)_{[i,\,n_i+1]} = \max\{A_{i\,n_i} + t_{[i+k,\,1]},\, C_{[i,\,n_i]}\} + t_{[i+k,\,1]},$$

$$C(\sigma_2)_{[i+1,\,j]} = C(\sigma_1)_{[i+1,\,j]} + t_{[i+k,\,1]}\ ,\ \text{for}\ \ j = 1, 2, \cdots, n_{i+1},$$

$$C(\sigma_2)_{[i+l,\,j]} = \max\{A_{i\,n_i} + t_{[i+1,\,n_{i+1}]}, \cdots, A_{i\,n_i} + t_{[i+l-1,\,n_{i+l-1}]}, A_{i\,n_i} + t_{[i+l,\,j]}, A_{i\,n_i} + t_{[i+k,\,1]}, C_{[i,\,n_i]}\}$$

$$+ \sum_{h=1}^{l} s_{[i+h]} + \sum_{h=1}^{l-1}\sum_{j=1}^{n_{i+h}} t_{[i+h,\,j]} + \sum_{h=1}^{j} t_{[i+l,\,h]} + t_{[i+k,\,1]},\quad \text{for}\ \ l = 2, \cdots, k-1,\ j = 1, 2, \cdots, n_{i+l},$$

$$C(\sigma_2)_{[i,\,n_i+1]} + \sum_{l=1}^{k-1}\sum_{j=1}^{n_{i+l}} C(\sigma_2)_{[i+l,\,j]} = C(\sigma_1)_{[i+k,\,1]} + \sum_{l=1}^{k-1}\sum_{j=1}^{n_{i+l}} C(\sigma_1)_{[i+l,\,j]} + (\max\{A_{i\,n_i} + t_{[i+k,\,1]},\, C_{[i,\,n_i]}\}$$

$$- \max\{A_{i\,n_i} + t_{[i+1,\,n_{i+1}]}, \cdots, A_{i\,n_i} + t_{[i+k-1,\,n_{i+k-1}]}, A_{i\,n_i} + t_{[i+k,\,1]}, C_{[i,\,n_i]}\})$$

$$+ (t_{[i+k,\,1]} \sum_{l=1}^{k-1} n_{i+l} - \sum_{h=1}^{k-1}\sum_{j=1}^{n_{h+i}} t_{[i+h,\,j]}) - \sum_{h=1}^{k} s_{[i+h]}, \quad (12)$$

$$C(\sigma_2)_{[i+k,\,1]} = \max\{A_{i\,n_i} + t_{[i+k,\,1]},\, A_{i\,n_i} + t_{[i+1,\,n_{i+1}]}, \cdots, A_{i\,n_i} + t_{[i+k-1,\,n_{i+k-1}]}, A_{i\,n_i} + t_{[i+k,\,2]}, C_{[i,\,n_i]}\}$$

$$+ \sum_{h=1}^{k} s_{[i+h]} + \sum_{h=1}^{k-1}\sum_{j=1}^{n_{i+h}} t_{[i+h,\,j]} + t_{[i+k,\,1]} + t_{[i+k,\,2]}. \quad (13)$$

From (11) and (13), considering that $t_{[i+k,1]} \le t_{[i+k,2]}$, we have $C(\sigma_1)_{[i+k,\,2]} = C(\sigma_2)_{[i+k,\,1]}$. So, the completion times of all the jobs after job [*i*+*k*, 1] in schedule σ_1 are the same as those in schedule σ_2.

Furthermore, from (12), taking into account that $t_{[i+k,\,1]} \le t_{[i+l,\,1]}$, $l = 1, 2, \cdots, k-1$, we have

$$C(\sigma_2)_{[i,\,n_i+1]} + \sum_{l=1}^{k-1}\sum_{j=1}^{n_{i+l}} C(\sigma_2)_{[i+l,\,j]} < C(\sigma_1)_{[i+k,\,1]} + \sum_{l=1}^{k-1}\sum_{j=1}^{n_{i+l}} C(\sigma_1)_{[i+l,\,j]}\ .$$

Therefore, the total flowtime of σ_2 is less than the total flowtime of σ_1.

ii) Similar to the proof of i). □

4. Algorithms

In the previous section, we have derived some properties of the optimal solutions for several special cases. Since Theorems 1 and 2 state the conditions for the optimal solutions for cases 3.1 and 3.2, respectively, it is easy to develop polynomial time solution algorithms for these two cases. In this section, we develop some heuristics and branch-and-bound algorithms for cases 3.3, 3.4 and 3.5.

4.1 Heuristic algorithms

Our heuristic algorithms use a construction procedure to find an initial schedule, and then improve the performance of the solutions according to the optimal properties stated in the various theorems in the previous section.

For $F2/M_1 \succ M_2, class/\sum C_{[i,j]}$, we construct the following algorithms.

Heuristic algorithm A1(HAA1)

Step 1. Let $\sigma_i = [job(i,1), job(i,2), \cdots, job(i,c_i)]$ be the sequence in ascending order of their processing times on machine 1, $i = 1, 2, \cdots, c$, and set $\beta = \Phi$ (empty set).

Step 2. Select a job $(i, 1)$ such that $s_{i1} + a_{i1} = \min\{s_{k1} + a_{k1} \mid k = 1, 2, \cdots, c\}$. Remove job $(i, 1)$ from σ_i and place it in the first position of β, and set $h = i$.

Step 3. If $\sigma_1 \cup \sigma_2 \cup \cdots \cup \sigma_c = \Phi$, go to step 4, else

If $\sigma_h \neq \Phi$ and $a_{hm} \leq s_{i1} + a_{ik}$, remove job (h, m) from σ_h and place it in the last position of β. Otherwise, remove job (i, k) from σ_i and place it in the last position of β, and set $h = I$, where job (h, m) is the first job in σ_h, and job (i, k) satisfies $s_{i1} + a_{ik} = \min\{s_{g1} + a_{gl} \mid$ job (g, l) is the first job in $\sigma_g, g \neq h\}$. Go to step 3.

Step 4. For sequence β, calculate $(s_{[i],1} + \sum_{j=1}^{n_i} a_{[i,j]})/n_i$ for its batches, and sequence all batches of β in ascending order of $(s_{[i],1} + \sum_{j=1}^{n_i} a_{[i,j]})/n_i$.

Step 5. For the two batches containing the jobs of the same class, move their jobs that satisfy the conditions for the optimal solution given in Theorem 5. Denote the obtained sequence as β.

Step 6. Repeat step 4 if necessary. Stop.

Heuristic algorithm A2 (HAA2)

This heuristic is similar to HAA1 except step 3. We rewrite step 3 as follows.

Step 3. If $\sigma_1 \cup \sigma_2 \cup \cdots \cup \sigma_c = \Phi$, go to step 4, else

If $\sigma_h \neq \Phi$ and $(a_{hm} + a_{h,m+1})/2 \le s_{i1} + a_{ik}$, remove jobs ($h$, m) and (h, m+1) from σ_h and place them in the last two positions of β. Otherwise, remove job (i, k) from σ_i and place it in the last position of β, and set $h = I$, where jobs (h, m) and (h, m+1) are the first two jobs in σ_h, and job (i, k) satisfies $s_{i1} + a_{ik} = \min\{s_{g1} + a_{gl} \mid$ job (g, l) is the first job in $\sigma_g, g \neq h\}$. Go to step 3.

For $F2/M_2 \succ M_1, class/\sum C_{[i,j]}$, we have the following algorithms.

Heuristic algorithm B1 (HAB1)

Step 1. Let $\sigma_i = [job(i, 1), job(i, 2), \cdots, job(i, c_i)]$ be the sequence in ascending order of their processing times on machine 2, $i = 1, 2, \cdots, c$, and set $\beta = \Phi$.

Step 2. Select a job (i, 1) such that $s_{i2} + b_{i1} = \min\{s_{k2} + b_{k1} \mid k = 1, 2, \cdots, c\}$. Remove job ($i$, 1) from σ_i and place it in the first position of β, and set $h = i$.

Step 3. If $\sigma_1 \cup \sigma_2 \cup \cdots \cup \sigma_c = \Phi$, go to step 4, else

If $\sigma_h \neq \Phi$ and $b_{hm} \leq s_{i2} + b_{ik}$, remove job ($h$, m) from σ_h and place it in the last position of β. Otherwise, remove job (i, k) from σ_i and place it in the last position of β, and set $h = I$, where job (h, m) is the first job in σ_h, and job (i, k) satisfies $s_{i2} + b_{ik} = \min\{s_{g2} + b_{gl} \mid$ job (g, l) is the first job in $\sigma_g, g \neq h\}$. Go to step 3.

Step 4. For sequence β, calculate $(s_{[i],2} + \sum_{j=1}^{n_i} b_{[i,j]})/n_i$ for its batches, and sequence all batches of β in ascending order of $(s_{[i],2} + \sum_{j=1}^{n_i} b_{[i,j]})/n_i$.

Step 5. For the two batches containing jobs of the same class, move their jobs that satisfy the conditions for the optimal solution given in Theorem 8. Denote the obtained sequence as β.

Step 6. Repeat step 4 if necessary. Stop.

Heuristic algorithm B2 (HAB2)

This heuristic is similar to HAB1 except step 3. We rewrite step 3 as follows.

Step 3. If $\sigma_1 \cup \sigma_2 \cup \cdots \cup \sigma_c = \Phi$, go to step 4, else

If $\sigma_h \neq \Phi$ and $(b_{hm} + b_{h,m+1})/2 \leq s_{i2} + b_{ik}$, remove jobs ($h$, m) and (h, m+1) from σ_h and place them in the last two positions of β. Otherwise, remove job (i, k) from σ_i and place it in the last position of β, and set $h = I$, where jobs (h, m) and (h, m+1) are the first two jobs in σ_h, and job (i, k) satisfies $s_{i2} + b_{ik} = \min\{s_{g2} + b_{gl} \mid$ job (g, l) is the first job in $\sigma_g, g \neq h\}$. Go to step 3.

For $F2/s_{i1} = s_{i2}, a_{ij} = b_{ij} = t_{ij}, class/\sum C_{[i,j]}$, we construct the following algorithms.

Heuristic algorithm C1 (HAC1)

Step 1. Let $\sigma_i = [job(i, 1), job(i, 2), \cdots, job(i, c_i)]$ be a sequence in ascending order of

their processing times on machine 1 or 2, $i = 1, 2, \cdots, c$, and set $\beta = \Phi$.

Step 2. Select a job (i, 1) such that $s_i + t_{i1} = \min\{s_k + t_{k1} \mid k = 1, 2, \cdots, c\}$. Remove job ($i$, 1) from σ_i and place it in the first position of β, and set $h = i$.

Step 3. If $\sigma_1 \cup \sigma_2 \cup \cdots \cup \sigma_c = \Phi$, go to step 4, else

If $\sigma_h \neq \Phi$ and $t_{hm} \leq s_i + t_{ik}$, remove job (h, m) from σ_h and place it in the last position of β. Otherwise, remove job (i, k) from σ_i and place it in the last position of β, and set $h = I$, where job (h, m) is the first job in σ_h, and job (i, k) satisfies $s_i + t_{ik} = \min\{s_g + t_{g\,l} \mid$ job (g, l) is the first job in $\sigma_g, g \neq h\}$. Go to step 3.

Step 4. For sequence β, interchange the adjacent pairs of batches to improve the objective function.

Step 5. For the two batches containing jobs of the same class, move their jobs that satisfy the conditions for the optimal solution given in Theorem 11. Stop.

Heuristic algorithm C2 (HAC2)

This heuristic is similar to HAC1 except step 3. We rewrite step 3 as follows.

Step 3. If $\sigma_1 \cup \sigma_2 \cup \cdots \cup \sigma_c = \Phi$, go to step 4, else

If $\sigma_h \neq \Phi$ and $(t_{hm} + t_{h,\,m+1}) / 2 \leq s_i + t_{ik}$, remove jobs ($h$, m) and (h, m+1) from σ_h and place them in the last two positions of β. Otherwise, remove job (i, k) from σ_i and place it in the last position of β, and set $h = I$, where jobs (h, m) and (h, m+1) are the first two jobs in σ_h, and job (i, k) satisfies $s_i + t_{ik} = \min\{s_g + t_{g\,l} \mid$ job (g, l) is the first job in $\sigma_g, g \neq h\}$. Go to step 3.

4.2 Branch-and-bound algorithms

For each of the above cases, we derive a lower bound that can be used to reduce the size of the search tree generated by the branch-and-bound procedure.

We assume that a partial sequence σ_1 has been determined, in which job (p, q) is the last job, and σ_2 is the complement of σ_1, which include α unscheduled jobs.

For case 3.3, in set σ_2, let a_{ij}^k $(k = 1, 2, \cdots, \alpha)$ denote the processing time of job (i, j) on machine 1, where job (i, j) is the kth job when being sequenced among these α jobs in increasing order of their processing times on machine 1. Let $n_i^l > 0$ ($l = 1, 2, \cdots, \beta$) denote the job number of class i in σ_2 (excluding class p), where l denotes the lth position in increasing order of s_{i1}^l / n_i^l, and $s_i^l{}_1$ denotes the setup time of these n_i^l jobs on machine 1. A lower bound for the case 3.3 can be expressed as follows:

$$LB1 = \sum_{(i,\, j)\in\sigma_1} C_{ij} + \sum_{l=1}^{\beta}(s_{i1}^l \sum_{k=l}^{\beta} n_i^k) + \alpha(C_{pq} - b_{pq}) + \sum_{k=1}^{\alpha}(\alpha - k + 1)a_{ij}^k + \sum_{(i,\, j)\in\sigma_2} b_{ij} \,. \tag{14}$$

For case 3.4, in set σ_2, let b_{ij}^k $(k = 1, 2, \cdots, \alpha)$ denote the processing time of job (i, j) on machine 2, where job (i, j) is the kth job when being sequenced among these α jobs in increasing order of their processing times on machine 2. Let $n_i^l > 0$ ($l = 1, 2, \cdots, \beta$) denote the job number of class i in σ_2 (excluding class p), where l denotes the lth position in increasing order of $s_i^l{}_2$ / n_i^l and $s_i^l{}_2$ denotes the setup time of these n_i^l jobs on machine 2. A lower bound for the case 3.4 can be expressed as follows:

$$LB2 = \sum_{(i,\, j)\in\sigma_1} C_{ij} + \sum_{l=1}^{\beta}(s_{i2}^l \sum_{k=l}^{\beta} n_i^k) + \alpha C_{pq} + \sum_{k=1}^{\alpha}(\alpha - k + 1)b_{ij}^k \,. \tag{15}$$

For case 3.5, in set σ_2, let t_{ij}^k $(k = 1, 2, \cdots, \alpha)$ denote the processing time of job (i, j) on machine 1 or 2, where job (i, j) is the kth job when being sequenced among these α jobs in increasing order of their processing times on machine 1 or 2. Let $n_i^l > 0$ ($l = 1, 2, \cdots, \beta$) denote the job number of class i in σ_2 (excluding class p), where l denotes the lth position in increasing order of s_i / n_i^l and s_i^l denotes the setup time of these n_i^l jobs on machine 1 or 2. A lower bound for the case 3.5 can be expressed as follows:

$$LB3 = \sum_{(i,\, j)\in\sigma_1} C_{ij} + \sum_{l=1}^{\beta}(s_i^l \sum_{k=l}^{\beta} n_i^k) + \alpha C_{pq} + \sum_{k=1}^{\alpha}(\alpha - k + 1)t_{ij}^k + \sum_{k=1}^{\alpha}\max\{A_{pq} + t_{ij}^k - C_{pq},\, 0\}. \tag{16}$$

The branch-and-bound algorithm for case 3.3 can be described as follows: At the root node of the branch-and-bound search tree, the best result generated by HAA1 and HAA2 is applied

as an initial upper bound. At each node of the branch-and-bound search tree, the optimal properties of Theorems 3 to 5 are used as pruning devices, and if the current lower bound is greater than the upper bound, this node is pruned.

The branch-and-bound algorithm for case 3.4 can be described as follows: At the root node of the branch-and-bound search tree, the best result generated by HAB1 and HAB2 is applied as an initial upper bound. At each node of the branch-and-bound search tree, the optimal properties of Theorems 6 to 8 are used as pruning devices, and if the current lower bound is greater than the upper bound, this node is pruned.

The branch-and-bound algorithm for case 3.5 can be described as follows: At the root node of the branch-and-bound search tree, the best result generated by HAC1 and HAC2 is applied as an initial upper bound. At each node of the branch-and-bound search tree, the optimal properties of Theorems 9 to 11 are used as pruning devices, and if the current lower bound is greater than the upper bound, this node is pruned.

5. Computational results

The measures of the effectiveness of the heuristic algorithms are the average relative error and maximal relative error from the optimal total flowtime, where the relative error is defined as: relative error (%) = [(heuristic total flowtime – optimal total flowtime) x 100]/[optimal total flowtime]. The branch-and-bound algorithms are evaluated with respect to the average CPU time in seconds and the average number of nodes searched. The procedures were coded in VB 5.0 and run on a Celeron 300 PC.

Fifty random instances were generated for each of feasible combinations of n = 20, 30 jobs and different numbers of job classes. The job processing times for case 3.3 on machines 1 and 2 were uniformly distributed integers between 50 and 100 and between 1 and 50, respectively. The job processing times for case 3.4 on machines 1 and 2 were uniformly distributed integers between 1 and 50, and between 50 and 100, respectively. The job processing times for case 3.5 were uniformly distributed integers between 1 and 100. The class setup times for case 3.3 on machine 1, and case 3.5 on machines 1 and 2 were uniformly distributed integers between 1 and 100. The class setup times for case 3.4 on machine 2 were uniformly distributed integers

between 50 and 100. The class setup times for case 3.3 on machine 2, and for case 3.4 on machine 1 were randomly generated integers according to the definitions of cases 3.3 and 3.4, respectively. We present a summary of our findings in Tables 1, 2 and 3.

Table 1 reports the performance of HAA1 and HAA2 of case 3.3. It is obvious that HAA1 is better than HAA2 except that for 30 jobs divided into 3 classes. We notice that the flowtime is made up of the job processing times and the job class setups. In general, the optimal objective is obtained through balancing these two items. Since HAA2 tends to produce more batches than HAA1, HAA2 is more likely to produce a better solution for the scheduling problem with fewer job classes. It seems that the mean relative errors for the two heuristics are independent of the number of jobs. The mean relative error for HAA1 decreases as the number of class decreases.

Table 1 also reveals that the branch-and-bound algorithm can find the optimal solutions for all the problems generated. Both the searched nodes and CPU time tend to decrease as the number of classes increases, which is indicated by the expression of the lower bound (14) as the more the jobs are divided into job classes, the tighter the lower bound becomes.

Table 2 reports the performance of HAB1 and HAB2 and the branch-and-bound algorithm for case 3.4, and Table 3 the performance of HAC1 and HAC2 and the branch-and-bound algorithm for case 3.5. The algorithms for cases 3.4 and 3.5 have similar characters as those of the algorithms for case 3.3.

Since the conditions for the optimal solution for case 3.5 are stricter than those for cases 3.3 and 3.4, it is evident that the branch-and-bound algorithm for case 3.5 searches more nodes and requires more CPU time than those for cases 3.3 and 3.4.

6. Conclusions

In this paper we have considered several special cases of the two-machine flowshop scheduling with job class setups and derived some optimal solution properties for these problems to minimize the total flowtime. We have discussed the use of these optimal properties to develop heuristic algorithms. We have also developed branch-and-bound algorithms for cases 3.3, 3.4 and 3.5, in which the best solution of two heuristic algorithms is used as an initial

upper bound, and the derived lower bounds and optimal properties are used to reduce the size of the search tree. We have conducted computational experiments to test these algorithms and the results demonstrate that all of the algorithms are very efficient and effective in solving the special cases under study.

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 two anonymous referees for their helpful comments on an earlier version of this paper.

References

B.-H. Ahn and J.-H. Hyun, "Single facility multi-class job scheduling", *Computers & Operations Research*, 1990, 17, 265-272.

A. Allahverdi, "Minimizing mean flowtime in a two-machine flowshop with sequence independent setup times", *Computers & Operations Research*, 2000, 27, 111-127.

P. Brucker, *Scheduling Algorithms*, Spring-Verlag, Berlin, 1995.

T.C.E. Cheng, J.N.D. Gupta and G.Q. Wang, "A review of flowshop scheduling with setup times", *Production and Operations Management*, 2000, 9, 262-282.

H.A.J. Crauwels, A.M.A. Hariri, C.N. Potts and L.N. Van Wassenhove, "Branch and bound algorithms for single-machine scheduling with batch set-up times to minimize total weighted completion time", *Annals of Operations Research*, 1998, 83, 59-76.

H.A.J. Crauwels, C.N. Potts and L.N. Van Wassenhove, "Local search heuristics for single machine scheduling with batch set-up times to minimize total weighted completion time", *Annals of Operations Research*, 1997, 70, 261-279.

T. Gonzalez and S. Sahni, "Flowshop and job shop schedules: complexity and approximation", *Operations Research*, 1978, 26, 36-52.

J.N.D. Gupta, "Optimal schedules for single facility with two job classes", *Computers & Operations Research*, 1984, 11, 409-413.

J.N.D. Gupta, "Single facility scheduling with multiple job classes", *European Journal of*

Operational Research, 1988, 8, 42-45.

J.C. Ho and J.N.D. Gupta, "Flowshop scheduling with dominant machines", *Computers & Operations Research*, 1995, 22, 237-246.

A.J. Mason and E.J. Anderson, "Minimizing flow time on a single machine with job classes and setup times", *Naval Research Logistics*, 1991, 38, 333-350.

C.N. Potts, "Scheduling two job classes on a single machine", *Computers & Operations Research*, 1991, 18, 411-415.

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

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.

S. Webster and K.R. Baker, "Scheduling groups of jobs on a single machine", *Operations Research*, 1995, 43, 692-704.

H.-S. Woo and D.-S. Yim, "A heuristic algorithm for mean flowtime objective in flowshop scheduling", *Computers & Operations Research*, 1998, 25, 175-182.

Table 1. Performance evaluation of HAA1, HAA2 and the branch-and-bound algorithm.

No. of jobs	No. of classes	HAA1		HAA2		Branch-and-bound	
		Mean(%)	Max(%)	Mean(%)	Max(%)	Nodes	Times
20	2	0.050	0.573	0.092	0.869	4179	25.95
	4	0.000	0.000	0.046	0.919	968	5.97
	5	0.000	0.000	0.000	0.000	30	0.22
	10	0.000	0.000	0.000	0.000	17	0.284
30	3	0.838	3.426	0.497	3.426	65945	1181.52
	5	0.017	0.330	0.060	0.129	2646	44.658
	6	0.000	0.000	0.284	4.947	37	0.759
	10	0.006	0.366	0.024	0.146	29	0.832

Table 2. Performance evaluation of HAB1, HAB2 and the branch-and-bound algorithm.

No. of jobs	No. of classes	HAB1		HAB2		Branch-and-bound	
		Mean(%)	Max(%)	Mean(%)	Max(%)	Nodes	Times
20	2	0.923	3.766	0.922	3.716	5614	36.227
	4	0.499	4.428	0.055	4.428	3481	19.538
	5	0.000	0.000	0.002	0.045	37	0.2265
	10	0.000	0.000	0.000	0.000	18	0.291
30	3	0.941	2.137	0.542	4.221	58723	1102.12
	5	0.018	0.356	0.044	0.701	9675	130.06
	6	0.004	0.078	0.948	6.221	613	8.192
	10	0.000	0.000	0.018	0.066	30	0.789

Table 3. Performance evaluation of HAC1, HAC2 and the branch-and-bound algorithm.

No. of jobs	No. of classes	HAC1		HAC2		Branch-and-bound	
		Mean(%)	Max(%)	Mean(%)	Max(%)	Nodes	Times
20	2	0.947	1.392	0.787	1.351	7042	57.45
	4	0.108	2.153	0.985	9.038	5433	44.45
	5	0.066	1.257	1.351	6.761	228	1.6185
	10	0.000	0.000	0.000	0.000	117	0.897
30	3	1.796	3.969	1.395	3.462	96536	1326.70
	5	0.534	4.900	0.476	2.980	14759	443.45
	6	0.000	0.000	0.532	1.432	110	2.861
	10	0.048	0.894	1.669	5.339	89	0.914