Volume 55 Numbers 1–2 1–15 January 2017
ISSN: 0020-7543
INTERNATIONAL JOURNAL OF
Production Research
55th ANNIVERSARY
Official Journal of the International Foundation for Production Research
Editor-in-Chief: Alexandre Dolgui
Taylor & Francis
Taylor & Francis Group

Minimizing Total Weighted Completion Time on Batch and Unary Machines with Incompatible Job Families

Journal:	*International Journal of Production Research*
Manuscript ID	TPRS-2017-IJPR-1071.R3
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	03-Mar-2018
Complete List of Authors:	Huang, Zewen; Peking University, Department of Industrial Engineering & Management

	Shi, Zhongshun; University of Wisconsin-Madison, Department of Industrial & Systems Engineering Shi, Leyuan; Peking University, Department of Industrial Engineering & Management; University of Wisconsin-Madison, Department of Industrial & Systems Engineering
Keywords:	BATCH SCHEDULING, HYBRID PRODUCTION SYSTEMS, FLOW SHOP SCHEDULING, DISCRETE OPTIMISATION
Keywords (user):	batch machine, incompatible job families, total weighted completion time, approximation algorithm, constraint programming

Response to Reviewers

TPRS-2017-IJPR-1071.R2

We thank the referees and Editors for their comments and suggestions that have greatly improved the paper. Based on these comments and suggestions, we have made careful modifications on the manuscript. All changes made to the text are in blue.

Responses to Reviewer 1:

Thanks for all the comments and suggestions which improve this paper greatly and help us understand the constraint programming better. Thanks for the patience and kindly help.

Responses to Reviewer 2:

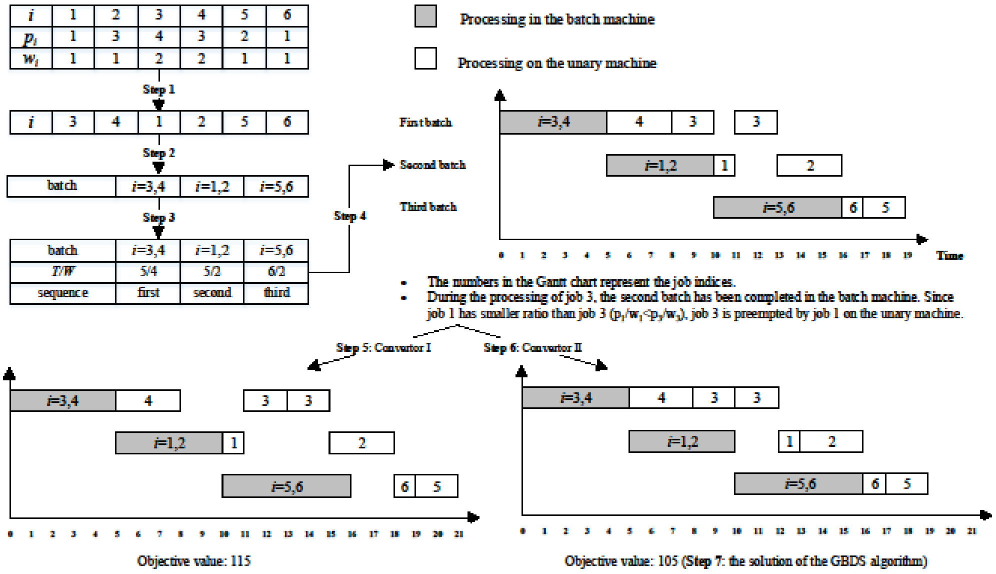
1. *The author should further define the considered problem clearly such as the sizes of all jobs are identical.*

   Thanks for this comment. We have added some descriptions to make the problem clear in the *Problem formulation* section.

   "*The batch machine has a fixed capacity. Only jobs from the same family can be assigned to the same batch. The sizes of all jobs are identical.*"
   "*The objective is to minimize the total weighted completion time.*"

2. *In order to the readers understand the proposed algorithms in this paper better, the authors should use an example to demonstrate the steps of the algorithms.*

   Thanks for this comment. We use an example to demonstrate the steps of the proposed algorithm as shown in Figure 1. In this example, there are six jobs which are grouped into two families. Detail contents have added in this revision.



i	1	2	3	4	5	6
p_i	1	3	4	3	2	1
w_i	1	1	2	2	1	1

i	3	4	1	2	5	6

batch	i=3,4	i=1,2	i=5,6

batch	i=3,4	i=1,2	i=5,6
T/W	5/4	5/2	6/2
sequence	first	second	third

- The numbers in the Gantt chart represent the job indices.
- During the processing of job 3, the second batch has been completed in the batch machine. Since job 1 has smaller ratio than job 3 ($p_1/w_1 < p_3/w_3$), job 3 is preempted by job 1 on the unary machine.

Fig 1. An example of the steps of the proposed GBDS algorithm.

Submitted to *International Journal of Production Research*

Minimizing Total Weighted Completion Time on Batch and Unary Machines with Incompatible Job Families

Zewen Huang[a], Zhongshun Shi[b*] and Leyuan Shi[a,b]

[a]Department of Industrial Engineering & Management, Peking University,Beijing,
100871, China

[b]Department of Industrial & Systems Engineering, University of Wisconsin-Madison,
WI, 53706, USA

[*]Corresponding author:

Dr. Zhongshun Shi

Department of Industrial & Systems Engineering

University of Wisconsin-Madison

WI, 53706, USA

Email: zhongshun.shi@wisc.edu

To appear in the *International Journal of Production Research*
Vol. 00, No. 00, 00 Month 20XX, 1–17

Minimizing Total Weighted Completion Time on Batch and Unary Machines with Incompatible Job Families

Zewen Huang[a], Zhongshun Shi[b*] and Leyuan Shi[a,b]

[a]*Department of Industrial Engineering & Management, Peking University, Beijing, 100871, China*
[b]*Department of Industrial & Systems Engineering, University of Wisconsin-Madison, WI, 53706, USA*

(*Received 00 Month 20XX; accepted 00 Month 20XX*)

This paper addresses the problem of scheduling on batch and unary machines with incompatible job families such that the total weighted completion time is minimized. A mixed-integer linear programming model is proposed to solve the problem to optimality for small instances. Tight lower bounds and a 4-approximation algorithm are developed. A constraint programming-based method is also proposed. Numerical results demonstrate that the proposed algorithms can obtain high quality solutions and have a competitive performance. Sensitivity analysis indicates that the performance of the proposed algorithms are also robust on different problem structures.

Keywords: batch machine; incompatible job families; total weighted completion time; approximation algorithm; constraint programming

1. Introduction

This paper focuses on a two-stage scheduling problem comprised of a batch machine in the upstream and a unary machine in the downstream. The batch machine can handle several jobs simultaneously up to its capacity. Once the process begins, the batch machine cannot be interrupted. Only when a batch is finished, the next batch can go into the batch machine. The unary machine processes jobs one by one. Incompatible job families are considered, which means that jobs from different families cannot be assigned to the same batch. The batch processing time depends on the related job family. For jobs in the same family, the batch processing times are identical. The objective is to find a schedule to minimize the total weighted completion time.

Batch machines have important applications in a variety of industrial systems, such as heat-treating ovens in the steel industry, burn-in operation in semiconductor final test and oxidation, diffusion in the wafer fabrication. In such industries, the batch machine usually means a bottleneck process, since it is often time-consuming and expensive. After batch processing, jobs are often needed to be processed one by one on the unary machine. This hybrid process with a batch machine and a unary machine is very common. We present two practical examples as follows. In the steel plant, the steel ingots are heated in the soaking pit before rolling process. The soaking pit is a batch machine which can reheat several steel ingots simultaneously. Then the steel ingots are rolled to a usable form of steel on the rough mill which can be regarded as a unary machine. In the semiconductor manufacturing factory, the oxidation area are batch machines. As to the downstream area of oxidation, such as etch area and photolithography area, machines are almost all unary machines.

We denote this two-stage scheduling problem as $\beta \to \delta|incompat|\sum w_jC_j$, where β, δ and

[*]Corresponding author. Email: zhongshun.shi@wisc.edu

incompat mean the batch machine, unary machine and incompatible job families, respectively. The third field specifies the objective function. Research on the two-stage scheduling problem that considering batch machines dates back to at least Ahmadi et al. (1992). They investigated the $\beta \to \delta$ problem and proposed a Full Batch-Dealing-Shortest Processing Time heuristic to minimize the total completion time $\sum C_j$. The authors proved that the worst case performance ratio of this heuristic is less than or equal to $1 + K/(2(K + 1))$, where K is the total number of batches.

Based on the work of Ahmadi et al. (1992), Hoogeveen and Velde (1998) used the concept of positional completion time to propose a Lagrangian lower bound which can be computed in $O(n \log n)$ time for the problem $\beta \to \delta | \sum C_j$. Potts, and Kovalyov (2000) reviewed the batch machine models considering a single batch machine, parallel batch machines, and shop problems with batch machines. Using the information invariance principle to generate a new selection rule, Kim and Kim (2002) proposed a genetic algorithm for the problem $\beta \to \delta | \sum C_j$. Su (2003) considered the limited waiting time constraints for the second stage and proposed a mixed integer linear programming model for the problem $\beta \to \delta$ to minimize the makespan C_{max}. Chung, Sun, and Liao (2017) and Huang et al. (2017) studied the two-stage scheduling problem with limited waiting time to minimize C_{max}. The release time and size of each job are considered.

All of the above reviewed research did not consider the incompatible job families. Due to the chemical incompatibility or different requirements for temperature, incompatible job families are important features in practical production. The batch machine with incompatible job families was early introduced in Uzsoy (1995) and was studied based on a single batch machine. Some efficient optimal algorithms were proposed to minimize the objective of C_{max}, maximum lateness L_{max} and the total weighed completion time $\sum w_j C_j$ on a single batch machine. Mehta and Uzsoy (1998) studied the problem of minimizing total tardiness on a single batch machine with incompatible job families. They presented a dynamic programming algorithm with polynomial time complexity when the number of jobs and machine capacity were fixed. Koh et al. (2005) considered the scheduling problem on a single batch machine with arbitrary job sizes and incompatible job families. The objective functions they considered are $C_{\max}$, $\sum C_j$ and $\sum w_j C_j$. Yao, Jiang, and Li (2012) considered a single batch machine scheduling problem with incompatible job families and dynamic job arrivals to minimize $\sum C_j$. A decomposed branch and bound algorithm was proposed. Dauzère-Pérès and Mönch (2013) studied a single batch machine with incompatible job families to minimize the weighted number of tardy jobs. They proposed two different mixed-integer linear programming models and a random key genetic algorithm to solve this scheduling problem. Cheng et al. (2014) considered the scheduling problem of a batch machine with incompatible job families. A mixed integer programming model and two polynomial time heuristics based on the longest processing time first rule were developed.

For the $\beta \to \delta$ configuration, the research on the batch machine with incompatible job families is limited. Koh, Kim, and Lee (2009) studied the problem $\beta \to \delta | incompat | C_{\max}$. They presented a mixed integer programming formulation for the problem. Some simple heuristic rules and genetic algorithm were proposed to solve the problem. Yao, Zhao, and Zhang (2012) studied $\beta \to \delta | C_{\max}$ problems with dynamic job arrivals and also considered the extensions with incompatible job families and limited waiting time. They presented TSEDD-based (time-symmetric earliest due date) algorithms for these problems. Fu, Sivakumar, and Li (2012) studied the problem of $\beta \to \delta | incompat | \overline{C}$ with a limited buffer, where $\overline{C}$ means the mean completion time. They presented a lower bound, two constructive algorithms and a differential evolution algorithm. Zhang et al. (2017) studied the problem $\beta \to \delta | incompat | \sum C_j$ with a limited buffer. An approximation algorithm and a hybrid differential evolution algorithm were proposed to solve the problem.

The problem $\beta \to \delta | incompat | \sum w_j C_j$ can be regarded as a special case of the general complex job-shop scheduling problems with batch machines (Knopp, Dauzère-Pérès, and Yugma 2017). However, the problem $\beta \to \delta | incompat | \sum w_j C_j$ has not been studied specially and no approximation algorithm has been developed for this problem. The goal of this research is to develop a constant factor approximation algorithm for this problem to fill this gap. In this paper, we first propose a mixed-integer linear programming model for problem $\beta \to \delta|incompat|\sum w_jC_j$. Based on the structure of this problem, three different lower bounds are developed. An approximation algorithm is proposed, and we prove that its worst-case performance ratio is bounded by the constant 4. A constraint programming-based method is also proposed. Numerical results show that the proposed algorithms have competitive and robust performances.

The remainder of this paper is organized as follows. Section 2 presents the problem definition and mathematical formulation. In Section 3, we develop three lower bounds for this problem. An approximation algorithm and the worst-case analysis are proposed in Section 4. In Section 5, a constraint programming-based method is developed. Computational experiments are conducted in Section 6, and we conclude this paper in Section 7.

2. Problem formulation

There are n jobs which are grouped into m incompatible families. In the first stage, these jobs are assigned to some batches to be processed on the batch machine. The batch machine has a fixed capacity. Only jobs from the same family can be assigned to the same batch. The sizes of all jobs are identical. In the second stage, the jobs are processed one by one in an arbitrary sequence on the unary machine. All the jobs are available at time zero. Preemption is not allowed on both the batch machine and unary machine. The objective is to minimize the total weighted completion time. For a better presentation, the notations and decision variables are illustrated as follows.

Notations

n : the number of jobs;
m : the number of families;
a : the number of batches;
N : the job set, where $N = \{1, 2, \cdots, n\}$;
F : the job family set, where $F = \{1, 2, \cdots, m\}$;
n_k : the number of jobs in family k;
B : the batch set, where $B = \{1, 2, \cdots, a\}$;
F_k : the set of jobs which belong to family k;
p_i : the processing time of job i on the unary machine;
w_i : the weight of job i.
q_k : the processing time of batches consisting of jobs of family k;
b : the batch capacity;
M_1 : a large positive number;
M_2 : a large positive number.

Variables

x_{il} : $x_{il} = 1$ if job i is assigned to batch l, and otherwise, $x_{il} = 0$;
y_{kl} : $y_{kl} = 1$ if batch l consists of the jobs of family k, and otherwise, $y_{kl} = 0$;
u_{ij} : $u_{ij} = 1$ if job i is processed before job j on the unary machine, and otherwise, $u_{ij} = 0$;
C_i : the completion time of job i on the unary machine.

Uzsoy (1995) proved the full batch property for a single batch machine under a regular measure of performance which is monotonically non-decreasing in the completion time of the individual job. Here, we prove that this full batch property can be extended to this two-stage problem in the following lemma.

Lemma 1. *There exists an optimal schedule of the problem* $\beta \to \delta|incompat|\sum w_jC_j$, *which consists of no partially full batches except possibly the last batch of each family.*

Proof. Let B_k^* denote the set of batches in which jobs belong to family k in an optimal schedule.

From the first batch to the penultimate batch in B_k^*, if one batch is not full, the jobs in the later batches can be inserted to this batch until it is full. After this exchange, the completion time of each job in family k on the batch machine is smaller than or equal to that in the optimal schedule. Thus we can keep the starting time of each job in family k on the unary machine unchanged. Repeating this inserting procedure, we can get a new schedule which consists of no partially full batches except possibly the last batch of each family. Since the starting time of each job on the unary machine in the new schedule can be the same as that in the optimal schedule, the new schedule is also an optimal schedule. □

Based on Lemma 1, the number of batches in family k is equal to $\lceil \frac{n_k}{b} \rceil$, where $\lceil \bullet \rceil$ means the rounding up function. Thus, we can set $a = \sum_{k \in F} \lceil \frac{n_k}{b} \rceil$ and this problem can be formulated as the following mixed-integer linear programming model which is denoted as BUMS (Batch and Unary Machine Scheduling) model.

$$
\begin{aligned}
(BUMS)\ \min & \sum_{j \in N} w_j C_j && && (1)\\
\text{s.t.} & \sum_{l \in B} x_{il} = 1, && \forall i \in N, && (2)\\
& \sum_{i \in N} x_{il} \leq b, && \forall l \in B, && (3)\\
& \sum_{k \in F} y_{kl} = 1, && \forall l \in B, && (4)\\
& x_{il} \leq y_{kl}, && \forall k \in F, i \in F_k, l \in B, && (5)\\
& u_{ij} + u_{ji} = 1, && \forall i, j \in N, i < j, && (6)\\
& C_i - p_i \geq C_j - M_1 u_{ij}, && \forall i, j \in N, i \neq j, && (7)\\
& C_i - p_i \geq \sum_{h=1}^{l} \sum_{k \in F} y_{kh} q_k - M_2(1 - x_{il}), && \forall i \in N, l \in B && (8)\\
& u_{ij} \in \{0,1\}, && \forall i, j \in N, i \neq j, && (9)\\
& y_{kl} \in \{0,1\}, && \forall k \in F, l \in B, && (10)\\
& x_{il} \in \{0,1\}, && \forall i \in N, l \in B. && (11)
\end{aligned}
$$

The objective function (1) is to minimize the total weighted completion time. Constraint (2) makes sure that one job can be assigned to only one batch. Constraint (3) guarantees all batches are full. By constraint (4), one batch only consists of jobs which belong to one family. Constraint (5) enforces that only when batch l consists of the jobs in family k, the jobs in family k can be assigned to batch l. Constraint (6) makes sure that jobs have processing precedence on the unary machine. Constraint (7) makes sure that only when a job finishes processing on the unary machine, the latter ones can start processing on the unary machine. Constraint (8) makes sure that job i can be processed on the unary machine only when the batch that job i belongs to finishes processing on the batch machine. Constraints (9), (10) and (11) are the range of the variables. Herein, we can set $M_1 = \sum_{k \in F} \lceil \frac{n_k}{b} \rceil q_k + \sum_{i \in N} p_i$ and $M_2 = \sum_{k \in F} \lceil \frac{n_k}{b} \rceil q_k$.

If we set the number of incompatible job families and all the weights to be 1, this problem reduces to problem $\beta \rightarrow \delta | \sum C_j$, which has been shown to be strongly NP-hard by Ahmadi et al. (1992). Thus the problem $\beta \rightarrow \delta | incompat | \sum w_j C_j$ is also NP-hard in strong sense.

3. Lower bounds

In this section, we develop three lower bounds for problem $\beta \to \delta|incompat|\sum w_jC_j$, which can be used to estimate the solution qualities of the proposed algorithms. Let t_j and r_l to denote the completion time of job j and batch l on the batch machine, respectively. Next, we first introduce the optimal Greedy Weighted Completion Time (GRWC) algorithm for the single batch scheduling problem with the objective to minimize total weighted completion time in the Lemma 2, and readers may refer to Uzsoy (1995) for details.

Lemma 2. *(Uzsoy (1995), GRWC) Index all jobs of the same family in non-increasing order of their weights w_i. For each family, starting from the first job, form full batches greedily. For the single batch scheduling problem with the objective to minimize total weighted completion time, the optimal schedule can be achieved by sequencing all batches in non-decreasing order of T_l/W_l, where T_l denotes the processing time of the batch l and W_l is the total weight of the jobs in the batch l.*

Definition 1. Γ_{GRWC} *is defined as the total weighted completion time achieved by the GRWC algorithm for the single batch scheduling problem.*

Considering the situation that all the jobs after the batch machine can be processed on the unary machine immediately, i.e., the waiting time on the unary machine is omitted, we derive the first lower bound denoted by LB_1 in the Theorem 1.

Theorem 1. *Form and sort the batches based on the GRWC algorithm. Then*

$$LB_1 = \Gamma_{\text{GRWC}} + \sum_{j=1}^{n} w_j p_j$$

is a lower bound of the problem $\beta \to \delta|incompat|\sum w_jC_j$.

Proof. Let (x^*, y^*, u^*, C^*) be an optimal solution of this problem. For the completion time C_j^* of job j, let w_j^* and p_j^* denote its related weight and processing time. Obviously, each C_j^* consists of three parts: the completion time t_j^* on the batch machine, the waiting time $\triangle_j^* \geq 0$ in the queue of the unary machine and the processing time p_j^* on the unary machine. Thus, in terms of the optimal objective value $\sum_{j=1}^{n} w_j^* C_j^*$, we have the following inequalities

$$\begin{aligned}\sum_{j=1}^{n} w_j^* C_j^* &= \sum_{j=1}^{n} w_j^*(t_j^* + \triangle_j^* + p_j^*)\\ &\geq \sum_{j=1}^{n} w_j^* t_j^* + \sum_{j=1}^{n} w_j^* p_j^*\\ &\geq \Gamma_{GRWC} + \sum_{j=1}^{n} w_j p_j = LB_1.\end{aligned}$$

Note that the last inequality holds according to Lemma 2, and this completes the proof. □

Definition 2. *Sort all the jobs in non-decreasing order of the ratios p_j/w_j for all $j \in N$ to form a WSPT (weighted shortest processing time) list. Denote the set of jobs whose positions are before job j in the WSPT list as Θ_j. Define Γ_{WSPT} as the total weighted completion time obtained by the WSPT rule for the single unary machine, i.e., $\Gamma_{\text{WSPT}} = \sum_{j\in N} w_j \sum_{i\in\Theta_j \bigcup\{j\}} p_i$.*

The WSPT rule gives the optimal solution for the single unary machine to minimize the total weighted completion time (Smith 1956). In Eastman, Even, and Isaacs (1964), the authors proposed a lower bound for the problem of scheduling n jobs on some identical unary machines with the objective to minimize the total weighted completion time. We introduce it in Lemma 3 and this helps us to develop another lower bound LB_2 in Theorem 2.

Lemma 3. *(Eastman, Even, and Isaacs 1964)* $\frac{1}{M}\Gamma_{\mathrm{WSPT}} + \frac{M-1}{2M}\sum_{i\in N} w_i p_i$ *is a lower bound for the problem of scheduling* n *jobs on* M *identical unary machines with the objective to minimize the total weighted completion time.*

Theorem 2. *Form and sort the batches based on the GRWC algorithm. Then*

$$LB_2 = \Gamma_{\mathrm{GRWC}} + \frac{1}{a}\Gamma_{\mathrm{WSPT}} + \frac{a-1}{2a}\sum_{i\in N} w_i p_i$$

is a lower bound of the problem $\beta \to \delta|incompat|\sum w_j C_j$.

Proof. Let (x^*, y^*, u^*, C^*) be an optimal solution of this problem. When one batch is finished on the batch machine, we let the jobs of this batch be processed immediately and consecutively on the unary machine without changing the relative positions in the optimal solution. Let C_j^0 denote the new completion time of job j. For each job j, it is easy to check that C_j^0 would be smaller than or equal to C_j^*. Thus we have the following inequalities

$$\begin{aligned}\sum_{j=1}^{n} w_j^* C_j^* &\geq \sum_{j=1}^{n} w_j^* C_j^0 \\ &= \sum_{l=1}^{a}\sum_{i=1}^{h_l^*} \lambda_{li}^* \mu_{li}^* + \sum_{l=1}^{a}\sum_{i=1}^{h_l^*} \lambda_{li}^* \sum_{j\leq i} \nu_{lj}^* \\ &\geq \Gamma_{GRWC} + \sum_{l=1}^{a}\sum_{i=1}^{h_l^*} \lambda_{li}^* \sum_{j\leq i} \nu_{lj}^*,\end{aligned}$$

where the completion time on the batch machine μ_{lj}^*, the processing time ν_{lj}^* and the weight λ_{lj}^* of job j assigned to batch l in the optimal solution are introduced. h_l^* denotes the number of jobs assigned to batch l in the optimal solution. In the problem of scheduling n jobs on a identical unary machines to minimize the total weighted completion time, we can construct a feasible solution, i.e., the l-th machine processes h_l^* jobs for $l = 1, \cdots, a$. Then the objective value of this feasible solution can be represented as the expression $\sum_{l=1}^{a}\sum_{i=1}^{h_l^*} \lambda_{li}^* \sum_{j\leq i} \nu_{lj}^*$. According to Lemma 3, the lower bound in Lemma 3 is also less than or equal to this expression. Thus, we have

$$\begin{aligned}\sum_{j=1}^{n} w_j^* C_j^* &\geq \Gamma_{GRWC} + \sum_{l=1}^{a}\sum_{i=1}^{h_l^*} \lambda_{li}^* \sum_{j\leq i} \nu_{lj}^* \\ &\geq \Gamma_{GRWC} + \frac{1}{a}\Gamma_{WSPT} + \frac{a-1}{2a}\sum_{i\in N} w_i p_i \\ &= LB_2.\end{aligned}$$

This completes the proof. □

Next, we take a contrary position to develop another lower bound, denoted by LB_3. Considering the situation that all the jobs are available on the unary machine after the first batch is finished at time r_1, this problem is relaxed to the single unary machine scheduling problem with identical release time r_1 to minimize the total weighted completion time. The relaxed problem can be solved to optimality by the WSPT rule.

Theorem 3. *Let q_0 denote the minimum value of q_k for $k \in F$. Then*

$$LB_3 = \sum_{j=1}^{n} w_j q_0 + \Gamma_{\text{WSPT}}$$

is a lower bound of the problem $\beta \to \delta|incompat|\sum w_j C_j$.

Proof. Let (x^*, y^*, u^*, C^*) be an optimal solution of this problem, and for the completion time C_j^* of job j, let w_j^* and p_j^* denote its related weight and processing time. Thus, in terms of the optimal objective value $\sum_{j=1}^{n} w_j^* C_j^*$, we have the following inequalities

$$\begin{aligned} \sum_{j=1}^{n} w_j^* C_j^* &\geq \sum_{j=1}^{n} w_j^* (r_1 + \sum_{i=1}^{j} p_i^*) \\ &= \sum_{j=1}^{n} w_j r_1 + \sum_{j=1}^{n} w_j^* \sum_{i=1}^{j} p_i^* \\ &\geq \sum_{j=1}^{n} w_j q_0 + \Gamma_{WSPT} = LB_3. \end{aligned}$$

□

The time complexities to calculate LB$_1$, LB$_2$ and LB$_3$ are all $O(n \log n)$ and can be calculated very fast. Based on the derivation process of these three lower bounds, their qualities depend heavily on the batch processing time and the sum of the processing time of jobs in one batch on the unary machine. When the batch processing time is small enough, LB$_3$ is tighter than LB$_1$ and LB$_2$. Under this situation, the batch processing time has little influence on the objective, and jobs tend to finish processing on the batch machine earlier. On the other hand, if the batch processing time is large enough, the jobs within one batch should be processed on the unary machine consecutively. This situation matches the structures of LB$_1$ and LB$_2$, and thus LB$_1$ and LB$_2$ are tighter than LB$_3$. The relationship between LB$_1$ and LB$_2$ depends on the data of processing time and weight. In this paper, we set the lower bound, LB, as the maximum value of LB$_1$, LB$_2$ and LB$_3$.

4. A 4-approximation algorithm

In this section, a preemptive algorithm is developed. When one batch is finished on the batch machine while the unary machine is busy, the job that is processed on the unary machine can be disrupted by other jobs. Based on the preemptive schedule achieved by running the preemptive algorithm, a 4-approximation algorithm is proposed to solve this problem. Both algorithms consist of two stages. The first stage is to determine how to form batches and the batch sequence processed on the batch machine. The second stage is to determine the job sequence processed on the unary machine. We denote the approximation algorithm as GBDS (Greedy Batch and Dynamic Selection).

4.1 *Preemptive GBDS algorithm*

We first introduce the preemptive GBDS algorithm. The preemption occurs only when one batch is finished on the batch machine while the unary machine is busy. It is described as follows.

Preemptive GBDS

```
Step 1. For each $k$, sort jobs of $F_k$ in a non-increasing order of $w_i,\ i \in F_k$;
Step 2. Form full batches greedily from the first job in each family;
Step 3. Calculate the value of $T_l/W_l$ for all batches, where $T_l$ denotes the processing time of the
        batch $l$ and $W_l$ is the total weight of the jobs in the batch $l$. The batches are processed on
        the batch machine in non-decreasing order of $T_l/W_l$;
Step 4. Set the current time $TNOW$ as the finished time of the first batch on the batch machine
        $r_1$.
Step 5. Denote the set of jobs which have finished processing on the batch machine and are
        waiting to be processed on the unary machine as $J_{TNOW}$ at the time $TNOW$. Select the job
        $i^* = arg\min_i p_i/w_i, i \in J_{TNOW}$,
          • If no batches are finished during the processing of job $i^*$, process job $i^*$ non-preemptively
            and set $TNOW = C_{i^*}$;
          • Else, denote the batch which are processing on the batch machine at the current time as
            $l^*$. Process job $i^*$ partially until the finished time of batch $l^*$. Set $TNOW = r_{l^*}$. Then
            stop processing job $i^*$ and return the remaining part of job $i^*$ with the criterion $p_{i^*}/w_{i^*}$
            to the set $J_{TNOW}$;
Step 6. Repeat Step 5 until all jobs are completed. Return the schedule $\pi$.
```

Figure 1 is an illustration of the preemptive GBDS algorithm. The horizontal axis represents the time. The ratios p/w for all the jobs satisfy the following inequalities: $p_{l1}/w_{l1} \le p_{l2}/w_{l2} \le p_{l3}/w_{l3}$ for batch $l = 1, 2, 3, 4$; $p_{23}/w_{23} \le p_{31}/w_{31}$; $p_{42}/w_{42} \le p_{32}/w_{32}$ and $p_{33}/w_{33} \le p_{43}/w_{43}$. Based on the above preemptive GBDS algorithm, it is easy to check that the job 23 should not be preempted, but the job 32 should be preempted. The gray boxes represent the preempted jobs. Next, we give a upper bound of the solution of the preemptive GBDS algorithm in Theorem 4.

Figure 1. An illustration of the preemptive GBDS algorithm.

Theorem 4. *Let π_P denote the schedule generated by the preemptive GBDS algorithm, and $Z(\pi_P)$ denote its corresponding objective value. The following equality holds*

$$Z(\pi_P) \le \Gamma_{\text{GRWC}} + \Gamma_{\text{WSPT}}.$$

Proof. For job j, the completion time on the unary machine C_j^P, the finished time on the batch machine t_j^P, the waiting time in the queue of the unary machine Δ_j^P in the solution of the preemptive GBDS algorithm are introduced. For each job j, we have

$$w_j C_j^P = w_j(t_j^P + \Delta_j^P + p_j).$$

The procedures of the preemptive GBDS algorithm can guarantee that when any batch is finished on the batch machine, the unary machine is available. Thus, the earliest start time of any job j is equal to its finished time on the batch machine t_j^P. Denote the set of jobs which have finished processing on the batch machine and are waiting to be processed on the unary machine during the time between t_j^P and the start time on the unary machine of job j as N_j. We have $N_1 \bigcup N_2 \bigcup \cdots \bigcup N_n = N$. Since the unary machine always selects job i with smallest ratio

$p_i/w_i, i \in N_j$ to process, thus the waiting time of any job j satisfies

$$\Delta_j^P \leq \sum_{i \in N_j \bigcap \Theta_j} p_i \leq \sum_{i \in \Theta_j} p_i,$$

where Θ_j is the set of jobs whose positions are before job j in the WSPT list. Therefore we have the following inequalities

$$\begin{aligned} Z(\pi_P) &= \sum_{j \in N} w_j(t_j^P + \Delta_j^P + p_j) \\ &\leq \sum_{j \in N} w_j(t_j^P + \sum_{i \in \Theta_j} p_i + p_j) \\ &= \sum_{j \in N} w_j t_j^P + \sum_{j \in N} w_j \sum_{i \in \Theta_j \bigcup \{j\}} p_i \\ &= \Gamma_{GRWC} + \Gamma_{WSPT}, \end{aligned}$$

This completes the proof. □

4.2 *GBDS algorithm*

The solution of the preemptive GBDS algorithm is not feasible for the problem $\beta \rightarrow \delta|incompat|\sum w_j C_j$, since preemption exists. Based on the preemptive solution, we derive the GBDS algorithm to obtain a feasible solution. The idea is to run the preemptive GBDS algorithm combining the GRWC algorithm and dynamic WSPT rule to get a preemptive schedule. The preemption is allowed only on the unary machine and when one batch is finished on the batch machine while the unary machine is busy. Then the preemptive schedule is converted to a non-preemptive schedule based on the sequence of the completion time of the last piece or first piece of jobs.

GBDS

Step 1. For each k, sort jobs of F_k in a non-increasing order of $w_i,\ i \in F_k$;
Step 2. Form full batches greedily from the first job in each family;
Step 3. Calculate the value of T_l/W_l for all batches, where T_l denotes the processing time of the batch l and W_l is the total weight of the jobs in the batch l. The batches are processed on the batch machine in non-decreasing order of T_l/W_l;
Step 4. Run the preemptive GBDS algorithm to get a preemptive schedule.
Step 5. Adjust the preemptive schedule based on Convertor I to get a feasible schedule.

- *Convertor I.* The jobs are processed on the unary machine continuously by non-decreasing order of the completion time of the last piece of jobs with the constraint that no job can start before its completion time on the batch machine.

Step 6. Adjust the preemptive schedule based on Convertor II to get another feasible schedule.

- *Convertor II.* The jobs are processed on the unary machine continuously by non-decreasing order of the completion time of the first piece of jobs with the constraint that no job can start before its completion time on the batch machine.

Step 7. Return the better schedule with smaller objective value.

In Figure 2, we give an example to illustrate the steps in the GBDS algorithm. In this example, there are six jobs ($i = 1, 2, \cdots, 6$). The capacity of the batch machine is set as 2. The first four jobs ($i = 1, 2, 3, 4$) are in a family and the related processing time in the batch machine is 5. The remaining jobs ($i = 5, 6$) are in another family and the related processing time in the batch machine is 6. In Step 4, the job 3 is preempted by job 1 on the unary machine. It is because the second batch has been completed in the batch machine during the processing of job 3 and job 1 has smaller ratio than job 3 ($p_1/w_1 < p_3/w_3$). As shown in Figure 2, the two convertors in Step 5 and 6 are used to adjust the preemptive schedule to feasible schedules.

Figure 2. An example of the steps in the GBDS algorithm.

In Theorem 5, we prove that the relationship between the preemptive GBDS and GBDS has the following result.

Theorem 5. *Let π_P and π_N denote the schedules generated by the preemptive GBDS and GBDS, respectively. Z is defined as the related objective function value. We have*

$$Z(\pi_N) \leq 2Z(\pi_P).$$

Proof. In the preemptive GBDS algorithm and the GBDS algorithm, the *Step 1* to *Step 3* are same and determine how to form batches and the sequence of batches on the batch machine. After *Step 3*, each job j has a finished time t_j on the batch machine, which can be regarded as the release time to the unary machine. So the sequence decisions on the unary machine are same with the problem $1|r_j|\sum w_jC_j$. Phillips, Stein, and Wein (1998) proved that given a preemptive schedule P for $1|r_j, pmtn|\sum w_jC_j$, the procedure of *Convertor I* in the GBDS algorithm produces a non-preemptive schedule N in which $C_j^N \leq 2C_j^P$ for each job j. Hence we have $Z(\pi_N) \leq 2Z(\pi_P)$, and this completes the proof. □

The time complexity of the GBDS algorithm is $O(n^2)$. Next, we prove that the worst-case performance ratio of the GBDS algorithm is bounded by 4 in Theorem 6.

Theorem 6. *The GBDS algorithm is a 4-approximation algorithm for problem $\beta \rightarrow \delta|incompat|\sum w_jC_j$.*

Proof. Let Z^*_{opt} denote the optimal objective value. Combining the results in Theorem 4 and Theorem 5, we have the following inequalities

$$\begin{aligned}
\frac{Z(\pi_N)}{Z^*_{opt}} &\leq \frac{2Z(\pi_P)}{Z^*_{opt}} \\
&\leq \frac{2(\Gamma_{GRWC} + \Gamma_{WSPT})}{Z^*_{opt}} \\
&= 2(1 + \frac{\Gamma_{GRWC} + \Gamma_{WSPT} - Z^*_{opt}}{Z^*_{opt}}) \\
&\leq 2(1 + \frac{\Gamma_{GRWC} + \Gamma_{WSPT} - LB_3}{LB_1}) \\
&= 2(1 + \frac{\Gamma_{GRWC} - \sum_{j=1}^n w_j q_0}{\Gamma_{GRWC} + \sum_{j=1}^n w_j p_j}) \\
&< 2(1 + 1) \\
&= 4.
\end{aligned}$$

This completes the proof. □

5. A constraint programming-based method

Constraint programming (CP), a method for solving the constraint satisfaction problem, now becomes a complementary solution technique for the combinatorial optimization problems and has a promising success for solving scheduling problems (Öztürk et al. 2013; Ham and Cakici 2016). The search algorithm of CP Optimizer is based on a self-adapting large neighbourhood search (Laborie and Godard 2007). In this section, a constraint programming-based method, CPWS (Constraint Programming with Warm Start), is developed to search the solution space further. In the CPWS method, the solution of the GBDS algorithm is attached as the initial solution of the CP technique. The CP technique has two advantages. One is that the natural formulation in CP is closer to the problem description and the scheduling problem can be built as a CP optimizer model easily. The other is that CP can find high-quality solutions in a short time for some scheduling and planning problems.

In CP, the interval variables are used as the decision variables to denote the tasks in scheduling or planning problems. Each interval variable has a start time, an end time and the given duration. The interval variables and some functions used for the problem $\beta \rightarrow \delta|incompat|\sum w_j C_j$ are defined as below:

- f_i, a parameter representing the family index which job i belongs to;
- $batch_i$, an interval variable representing the processing task of job $i \in N$ in the batch machine;
- $unary_i$, an interval variable representing the processing task of job $i \in N$ on the unary machine;
- $state^{batching}$, state of batch machine representing a batch. This state function is used as a restriction on that only jobs from the same family can be processed simultaneously. A set of non-overlapping intervals, each maintaining a particular nonnegative integer value f_i, are indicated by state.
- $cumul^{capacity} = \sum_{i \in N} pulse(batch_i, 1)$. Each assignment of job to the batch machine increases the cumulative function at the start of the usage of batch machine, and decreases the function when the batch machine is released at its end time.

With the variables and functions defined above, we can reformulate the problem $\beta \rightarrow \delta|incompat|\sum w_j C_j$ via CP as follows:

$$\min \sum_{i \in N} w_i \cdot \text{endOf}(unary_i) \tag{12}$$

$$\text{s.t.} \quad \text{endBeforeStart}(batch_i, unary_i), \quad \forall i \in N, \tag{13}$$

$$\text{alwaysEqual}(state^{batching}, batch_i, f_i), \forall i \in N, \tag{14}$$

$$\text{noOverlap}(unary), \tag{15}$$

$$cumul^{capacity} \leq b. \tag{16}$$

The objective (12) is to minimize the total weighted completion time. Constraint (13) ensures the precedence relation between the batch machine and the unary machine for each job. Constraint (14) specifies that the jobs in the same batch have the same start and end time in the batch machine. Constraint (15) imposes that the jobs on the unary machine should be processed one by one. Constraint (16) ensures that the total number of jobs in one batch cannot exceed the batch capacity.

The CPWS method integrates the solution of the GBDS algorithm as a warm start, then the CP solver with default configuration is used to solve the CP model. The pure CP method without warm start is also used to solve the problem. The time limits for the CPWS method and the pure CP method are set as 5 minutes.

6. Computational experiments

This section presents results of the computational experiments to test the performances of the proposed formulation and algorithm. All runs are made on a 64-bit Window 10 platform with Intel Core 3.2GHz CPUs and 8.0GB RAM. The mathematical formulation BUMS is solved by Cplex 12.7.1 with a time limit 3600 seconds under default configuration. The CP method is coded with OPL language in CP Optimizer 12.7.1.

6.1 *Problem instances generation*

We use the combination "n-m-b" to present the problem case, where n is the number of jobs, m is the number of job families and b is the batch capacity. As used by Ahmadi et al. (1992) and Koh et al. (2005), we present the instance generation rule below.

For each combination "n-m-b", set $n_k = b\lfloor \frac{n}{mb} \rfloor, k = 1, 2, \cdots, m-1$, and $n_m = n - \sum_{k=1}^{m-1} n_k$. For each job j, the weight w_j is randomly generated from the uniform distribution $[1, 2]$, and the processing time p_j on the unary machine is randomly generated from the uniform distribution $[1, 10]$. The batch processing time in family k, q_k, is randomly generated from the uniform distribution $[b(\sum_{j=1}^{n} p_j)/n - m, b(\sum_{j=1}^{n} p_j)/n + m]$, which can avoid the situation that the objective value is dominated by the batch machine or the unary machine. Let G_1 and G_2 denote two different sizes of instances, respectively. The configuration of each group is given as follows

G_1**:** $n = \{8, 12, 16, 20\}$, $m = \{2, 4\}$ and $b = \{2, 4\}$;
G_2**:** $n = \{50, 100, 200, 400\}$, $m = \{2, 5, 10\}$ and $b = \{10, 20\}$.

We use an additional requirement $\frac{n}{mb} \geq 1$ to get rid of some simple combinations. For G_1, we generate 10 instances for each combination to test the problem size that can be solved to optimality by the formulation BUMS. For G_2, we generate 50 instances for each combination to test the performance of the proposed algorithm. Thus, a total number of 1090 instances are used to test the performances of the proposed algorithm.

6.2 *Comparison with optimal solutions*

We present the numerical results based on the instances in G_1. For a convenient table size, we use the following notations to present the numerical results:

n-m-b: the combination of numbers of jobs and families, and batch capacity;
#Opt: the number of instances solved to optimality within one hour by the BUMS formulation;
Time: the average time in seconds to solve the instances to optimality by the BUMS formulation;
Gap-GBDS: the average gap between GBDS and optimal solutions;
Gap-CPWS: the average gap between CPWS and optimal solutions;
Gap-LB: the average gap between LB and optimal solutions.

In Table 1, the columns of *Time*, *Gap-GBDS* and *Gap-LB* report the average results for the instances that are solved to optimality in each combination. When no instance in one combination is solved to optimality within one hour by the BUMS formulation, the *#Opt* is zero and the columns of *Time*, *Gap-GBDS* and *Gap-LB* are displayed as "-". As shown in Table 1, the problem size that can be solved to optimality by BUMS is reported as 12 jobs within one hour on a single PC. When $n \geq 16$, only some instances are solved to optimality within one hour. The values of *Gap-GBDS* are all very small. The maximum average value of *Gap-GBDS* is 3.49% for the combination 12-2-2, which indicates that the proposed GBDS algorithm is capable of obtaining the near-optimal solutions for small-scale instances. Moreover, the proposed lower bound is also very tight to the optimal objective value. The maximum average value of *Gap-LB* appearing in the combination

Table 1. Comparison results of GBDS and LB with the optimal solutions of instances in G_1.

n-m-b	*#Opt*	*Time (s)*	*Gap-GBDS (%)*	*Gap-CPWS (%)*	*Gap-LB (%)*
8-2-2	10	0.28	2.16	0.00	4.55
8-2-4	10	0.19	0.59	0.00	1.82
8-4-2	10	0.13	2.15	0.00	4.81
12-2-2	10	70.56	3.49	0.00	4.06
12-2-4	10	6.40	0.46	0.00	2.56
12-4-2	10	2.60	2.99	0.00	3.66
16-2-2	1	501.08	0.60	0.00	1.42
16-2-4	9	517.09	1.68	0.00	2.24
16-4-2	4	1869.06	1.60	0.00	2.73
16-4-4	10	13.73	1.73	0.00	2.24
20-2-2	0	-	-	-	-
20-2-4	0	-	-	-	-
20-4-2	0	-	-	-	-
20-4-4	4	1495.50	1.54	0.00	1.60

8-4-2 is 4.81%. The *Gap-CPWS* column shows that the CPWS method can obtain the optimal solutions in a short time, which indicates the advantage of the CP technique.

6.3 *Comparison results for instances in G_2*

Table 2 presents detailed comparison results for instances in G_2. Here we use CP to denote the CP method without warm start. *Gap-GBDS*, *Gap-CPWS* and *Gap-CP* are the average values of gaps between the results achieved by the related algorithms and the lower bound. We calculate the gap between the *Algorithm* and *LB* as 100(*Obj(Algorithm)*-*LB*)/*LB* for each instance, then get the average value of gap among the 50 instances for each combination, where *Obj(Algorithm)* is the objective value achieved by the related *Algorithm* (GBDS, CPWS or CP). *Time* is the computing time in seconds for the GBDS algorithm. From the values of *Gap-GBDS* in Table 2, we can see that all the values are less than 4%. The maximum value of *Gap-GBDS* is 3.10% in the combination 100-10-10. The maximum average computing time is 2.83 seconds for the GBDS algorithm. These results indicate that the GBDS algorithm can obtain high quality solutions in a short time. When compared to CP, the CP method outperforms the GBDS algorithm in most combinations from the values of gap. When the solutions of the GBDS algorithm are integrated into the CP technique, the *Gap-CPWS* column shows that the CPWS method seems to yield the best results among these methods.

To further justify this observation, we establish the hypothesises as follows:

- GBDS-CP (H_0: $Gap\text{-}GBDS - Gap\text{-}CP \geq 0$ and H_1: $Gap\text{-}GBDS - Gap\text{-}CP < 0$)
- CPWS-CP (H_0: $Gap\text{-}CPWS - Gap\text{-}CP \geq 0$ and H_1: $Gap\text{-}CPWS - Gap\text{-}CP < 0$)

In Table 2, the columns of *GBDS-CP* and *CPWS-CP* represent the p-values obtained by the T-test when comparing the related two algorithms. When comparing GBDS and CP, the *GBDS-CP* column shows that the CP method is better than the GBDS algorithm in most combinations. But when the solutions of the GBDS algorithm are attached as the initial solutions of the CP technique, the *CPWS-CP* column shows that the results of the CP technique can be improved. These results indicate that the GBDS algorithm can provide high-quality solutions and the CPWS algorithm has a competitive performance.

6.4 *Sensitivity analysis*

The relationship between the batch processing time and the sum of the processing time of jobs in one batch on the unary machine has an effect on the problem structure. To test the robustness of the GBDS algorithm, we reduce and enlarge the batch processing time as $0.5q_k$ and $1.5q_k$ for each

Table 2. Comparison results for instances in G_2.

n-m-b	*Time (s)*	*Gap-GBDS (%)*	*Gap-CPWS (%)*	*Gap-CP (%)*	*GBDS-CP*	*CPWS-CP*
50-2-10	0.27	2.44	**1.29**	1.38	1.00E-00	2.71E-02
50-5-10	0.26	2.12	**1.48**	1.57	1.00E-00	3.18E-02
100-2-10	0.52	2.44	**1.11**	1.17	1.00E-00	2.98E-02
100-2-20	0.50	1.46	**1.05**	1.10	1.00E-00	5.35E-02
100-5-10	0.52	2.26	**1.50**	1.56	1.00E-00	1.48E-01
100-5-20	0.50	1.25	**1.01**	1.25	4.82E-01	2.16E-05
100-10-10	0.52	3.10	**2.24**	2.34	1.00E-00	4.41E-02
200-2-10	1.18	2.18	**0.84**	0.98	1.00E-00	1.35E-07
200-2-20	1.12	1.60	**0.85**	1.07	1.00E-00	4.00E-07
200-5-10	1.17	2.07	**1.29**	1.45	1.00E-00	4.25E-05
200-5-20	1.12	1.59	**1.34**	1.46	9.91E-01	5.08E-03
200-10-10	1.18	2.52	**1.96**	2.05	1.00E-00	7.43E-02
200-10-20	1.12	1.37	**1.11**	1.46	7.60E-02	4.20E-08
400-2-10	2.77	1.80	**0.60**	0.78	1.00E-00	8.89E-09
400-2-20	2.65	1.39	**0.64**	0.81	1.00E-00	1.13E-08
400-5-10	2.83	1.83	**0.96**	1.28	1.00E-00	3.79E-12
400-5-20	2.56	1.36	**1.07**	1.23	1.00E-00	1.51E-06
400-10-10	2.82	2.73	**2.12**	2.58	9.58E-01	9.65E-11
400-10-20	2.76	1.47	**1.35**	1.82	3.87E-06	3.35E-09

Note: The time limit for CPWS and CP is 5 minutes.
The bold font represents the best gap.

Table 3. Comparison results for instances in G_2 with reduced batch processing time.

n-m-b	*Time (s)*	*Gap-GBDS (%)*	*Gap-CPWS (%)*	*Gap-CP (%)*	*GBDS-CP*	*CPWS-CP*
50-2-10	0.25	5.59	**1.48**	1.53	1.00E-00	1.80E-01
50-5-10	0.23	7.98	**6.69**	6.95	1.00E-00	3.40E-02
100-2-10	0.47	6.12	**1.18**	1.51	1.00E-00	9.67E-08
100-2-20	0.44	4.75	**1.14**	1.31	1.00E-00	8.27E-03
100-5-10	0.47	7.19	**2.65**	2.87	1.00E-00	1.47E-04
100-5-20	0.45	7.66	**7.19**	7.86	4.95E-02	1.44E-07
100-10-10	0.48	9.66	**8.12**	8.48	1.00E-00	3.00E-04
200-2-10	1.05	6.97	**1.29**	2.25	1.00E-00	1.86E-22
200-2-20	1.03	6.10	**1.25**	2.24	1.00E-00	1.38E-19
200-5-10	1.11	7.07	**1.86**	2.90	1.00E-00	3.95E-17
200-5-20	1.05	6.89	**3.11**	3.84	1.00E-00	6.37E-07
200-10-10	1.17	7.86	**3.59**	4.11	1.00E-00	1.45E-05
200-10-20	1.02	9.71	**9.11**	10.29	7.21E-09	2.50E-19
400-2-10	2.74	7.42	**1.94**	6.89	9.79E-01	3.41E-25
400-2-20	2.64	6.88	**1.86**	6.52	9.17E-01	3.48E-24
400-5-10	2.79	7.31	**2.58**	7.86	3.16E-02	4.88E-24
400-5-20	2.54	6.90	**2.85**	8.23	2.97E-06	1.87E-24
400-10-10	2.64	7.49	**4.08**	9.00	4.23E-10	1.43E-29
400-10-20	2.45	7.80	**4.89**	11.12	2.28E-24	4.64E-33

Note: The time limit for CPWS and CP is 5 minutes.
The bold font represents the best gap.

family k, respectively. The results are reported in Table 3 and Table 4. The columns are same with that in Table 2.

When the batch processing time is reduced, the batches are finished earlier and more batches wait to be processed on the unary machine. In Table 3, the values of *Gap-GBDS* for all the combinations are small, where the maximum value of *Gap-GBDS* is 9.71% in the combination 200-10-20. When the numbers of jobs and families are large, the proposed GBDS algorithm can obtain better solutions than those of the CP method. As shown in the column *Gap-CPWS*, when the solutions of GBDS algorithm are integrated as a warm start of the CP technique, the performance of the CP technique can be improved a lot.

Table 4. Comparison results for instances in G_2 with enlarged batch processing time.

n-m-b	*Time (s)*	*Gap-GBDS (%)*	*Gap-CPWS (%)*	*Gap-CP (%)*	*GBDS-CP*	*CPWS-CP*
50-2-10	0.23	0.44	**0.41**	0.42	1.00E-00	9.00E-02
50-5-10	0.23	**0.25**	**0.25**	0.27	5.14E-04	5.17E-04
100-2-10	0.49	0.30	**0.28**	0.30	3.78E-01	1.44E-04
100-2-20	0.50	**0.33**	**0.33**	0.41	9.05E-04	6.12E-04
100-5-10	0.48	**0.26**	**0.26**	0.28	1.33E-04	1.16E-05
100-5-20	0.49	**0.12**	**0.12**	0.30	2.52E-08	2.52E-08
100-10-10	0.48	0.17	**0.16**	0.19	5.20E-06	7.12E-07
200-2-10	1.09	0.17	**0.16**	0.21	5.26E-14	9.10E-17
200-2-20	1.06	**0.22**	**0.22**	0.29	3.94E-14	3.35E-14
200-5-10	1.06	**0.16**	**0.16**	0.20	4.97E-11	4.87E-11
200-5-20	1.00	**0.19**	**0.19**	0.26	4.05E-08	3.84E-08
200-10-10	1.02	**0.15**	**0.15**	0.23	5.50E-10	3.38E-10
200-10-20	1.02	**0.08**	**0.08**	0.30	1.13E-14	1.13E-14
400-2-10	2.42	0.09	**0.08**	0.15	1.63E-15	6.18E-17
400-2-20	2.39	**0.12**	**0.12**	0.24	2.68E-18	1.59E-18
400-5-10	2.41	**0.09**	**0.09**	0.20	8.90E-20	8.50E-20
400-5-20	2.38	**0.12**	**0.12**	0.21	1.04E-15	1.03E-15
400-10-10	2.30	**0.09**	**0.09**	0.30	6.00E-22	7.19E-22
400-10-20	2.32	**0.10**	**0.10**	0.31	2.11E-12	2.11E-12

Note: The time limit for CPWS and CP is 5 minutes.
The bold font represents the best gap.

When the batch processing time is enlarged, the jobs in one batch tend to be processed on the unary machine consecutively. As shown in Table 4, all the values of gap between the GBDS and lower bound are very small. The maximum value of *Gap-GBDS* is 0.44% in the combination 50-2-10. These results show that the GBDS algorithm can obtain near-optimal solutions when the batch processing time is large. When comparing the GBDS algorithm and the CP method, most of the p-values are less than 5%, which indicates that the proposed GBDS algorithm has a competitive performance when enlarging the batch processing time. Since the solutions of the GBDS algorithm is good enough, most of the solutions of the CPWS algorithm is same with those of the GBDS algorithm.

The performances of LB_1, LB_2 and LB_3 also depend on the relationship between the batch processing time and the sum of the processing time of jobs in one batch on the unary machine. In Figure 3, LB_3 is tighter than LB_1 and LB_2 for all instances with reduced batch processing time. This is reasonable since the batch processing time is small. When processing one batch of jobs on the unary machine, many other batches have already been completed. This is very close to the structure of LB_3. However, when we enlarge the batch processing time, Figure 3 shows that LB_1 and LB_2 are tighter than LB_3. When the batch processing time are all larger than the sum of the processing time of jobs in one batch on the unary machine, the unary machine will process jobs in one batch consecutively and the waiting time become small, which leads that LB_1 and LB_2 are tighter than LB_3.

Figure 3. Performances of different lower bounds.

7. Conclusions

In this paper, we study the two-stage scheduling problem with a batch machine followed by a unary machine such that the total weighted completion time is minimized. The incompatible job families are considered. The problem is formulated as a mixed-integer linear programming model. Based on the structure of this problem, three lower bounds are developed. An approximation algorithm is proposed and we prove that the worst-case performance ratio of this algorithm is bounded by the constant 4. A constraint programming-based method is also developed. Numerical experiments are conducted to test the efficiency of the proposed algorithms. When compared to the optimal solutions and lower bounds, the proposed algorithms can obtain high quality solutions. When compared to the CP method, the proposed algorithms can provide good solutions and has a competitive performance. Sensitivity analysis indicates that the proposed algorithms have a robust computational performance for different problem structures.

Considering the numerical results on the approximation ratio performance are good, whether there exists a smaller constant factor for the proposed algorithm is an open question. In practice, the machines and operations are more complex. How to extend the lower bounds and approximation algorithm to the general problems with parallel machines or more than two operations is an important future work. Some general constraints, such as release dates, set-up time and re-entrant flows, should also be addressed in the future.

Acknowledgements

We thank the referees and Editors for their comments and suggestions that have greatly improved the paper. This research was supported in part by National Science Foundation of China [grant numbers 71690232] and by National Science Foundation [grant numbers CMMI-1435800].

References

Ahmadi, J. H., R. H. Ahmadi, S. Dasu, and C. S. Tang. 1992. "Batching and scheduling jobs on batch and discrete processors." *Operations research* 40 (4): 750–763.

Cheng, B., J. Cai, S. Yang, and X. Hu. 2014. "Algorithms for scheduling incompatible job families on single batching machine with limited capacity." *Computers & Industrial Engineering* 75: 116–120.

Chung, T. P., H. Sun, and C. J. Liao. 2017. "Two new approaches for a two-stage hybrid flowshop problem with a single batch processing machine under waiting time constraint." *Computers & Industrial Engineering* 113: 859–870.

Dauzère-Pérès, S., and L. Mönch. 2013. "Scheduling jobs on a single batch processing machine with incompatible job families and weighted number of tardy jobs objective." *Computers & Operations Research* 40 (5): 1224–1233.

Eastman, W. L., S. Even, and I. M. Isaacs. 1964. "Bounds for the optimal scheduling of n jobs on m processors." *Management science* 11 (2): 268–279.

Fu, Q., A. I. Sivakumar, and K. Li. 2012. "Optimisation of flow-shop scheduling with batch processor and limited buffer." *International Journal of Production Research* 50 (8): 2267–2285.

Ham, A. M., and E. Cakici. 2016. "Flexible job shop scheduling problem with parallel batch processing machines: MIP and CP approaches." *Computers & Industrial Engineering* 102: 160–165.

Hoogeveen, H., and S. Velde. 1998. "Scheduling by positional completion times: Analysis of a two-stage flow shop problem with a batching machine." *Mathematical Programming* 82: 273–289.

Huang, Z., Z. Shi, C. Zhang, and L. Shi. 2017. "A Note on Two new approaches for a two-stage hybrid flowshop problem with a single batch processing machine under waiting time constraint." *Computers & Industrial Engineering*, 110: 590–593.

Kim, B., and S. Kim. 2002. "Application of genetic algorithms for scheduling batch-discrete production system." *Production Planning & Control* 13 (2): 155–165.

Knopp, S., S. Dauzère-Pérès, and C. Yugma. 2017. "A batch-oblivious approach for complex job-shop scheduling problems." *European Journal of Operational Research* 263: 50–61.

Koh, S. G., P. H. Koo, D. C. Kim, and W. S. Hur. 2005. "Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families." *International Journal of Production Economics* 98 (1): 81–96.

Koh, S., Y. Kim, and W. Lee. 2009. "Scheduling two-machine flow shop with a batch processing machine." *International Conference on Computer & Industrial Engineering*, Troyes, France, July 6–9.

Laborie, P., and D. Godard. 2007. "Self-Adapting Large Neighborhood Search: Application to Single-mode Scheduling Problems." *Multidisciplinary International Conference on Scheduling: Theory and Applications*, Paris, France, August 28–31.

Mehta, S. V., and R. Uzsoy. 1998. "Minimizing total tardiness on a batch processing machine with incompatible job families." *IIE transactions* 30 (2): 165–178.

Öztürk, C., S. Tunali, B. Hnich and M. A. Örnek. 2013. "Balancing and scheduling of flexible mixed model assembly lines." *Constraints* 18 (3): 434–469.

Phillips, C., C. Stein, and J. Wein. 1998. "Minimizing average completion time in the presence of release dates." *Mathematical Programming* 82: 199–223.

Potts, C. N., and M. Y. Kovalyov. 2000. "Scheduling with batching: A review." *European Journal of Operational Research* 120 (2): 228–249.

Smith, W. E. 1956. "Various optimizers for single-stage production." *Naval Research Logistics* 3 (1–2): 59–66.

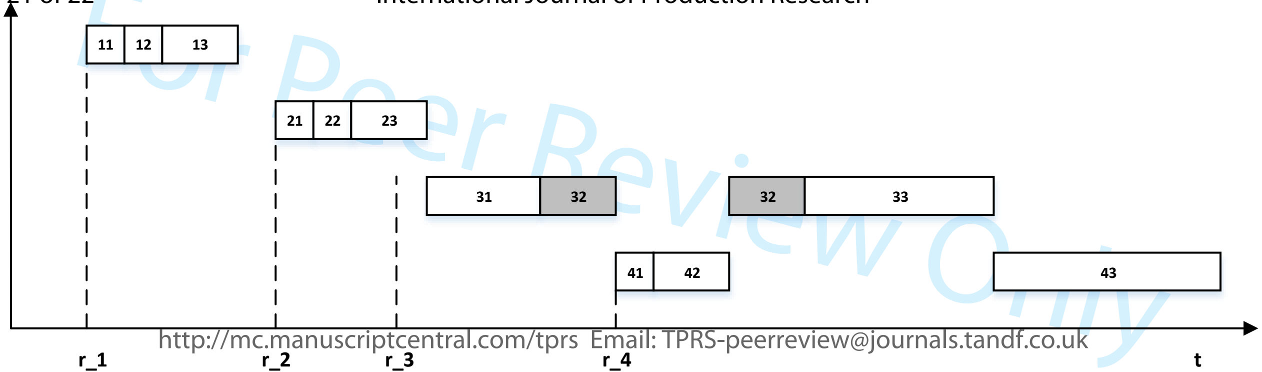
Su, L. H. 2003. "A hybrid two-stage flowshop with limited waiting time constraints." *Computers & Industrial Engineering* 44 (3): 409–424.

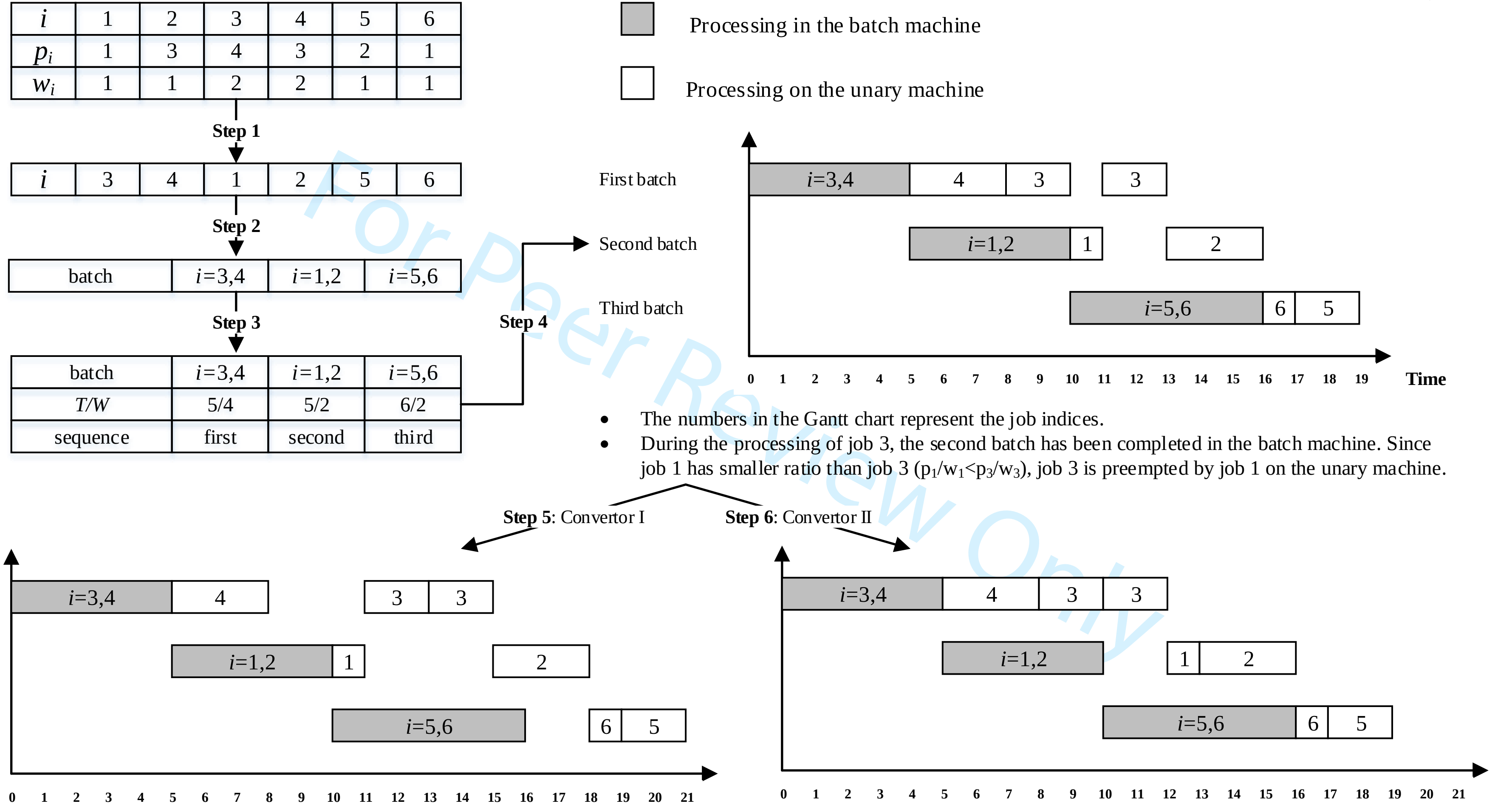
Uzsoy, R. 1995. "Scheduling batch processing machines with incompatible job families." *International Journal of Production Research* 33 (10): 2685–2708.

Yao, F. S., M. Zhao, and H. Zhang. 2012. "Two-stage hybrid flow shop scheduling with dynamic job arrivals." *Computers & Operations Research* 39 (7): 1701–1712.

Yao, S., Z. Jiang, and N. Li. 2012. "A branch and bound algorithm for minimizing total completion time on a single batch machine with incompatible job families and dynamic arrivals." *Computers & Operations Research* 39 (5): 939–951.

Zhang, C., Z. Shi, Z. Huang, Y. Wu, and L. Shi. 2017. "Flow shop scheduling with a batch processor and limited buffer." *International Journal of Production Research* 55 (11): 3217–3233.


11
12
13
21
22
23
31
32
32
33
41
42
43
r_1
r_2
r_3
r_4
t



i	1	2	3	4	5	6
p_i	1	3	4	3	2	1
w_i	1	1	2	2	1	1

Step 1

i	3	4	1	2	5	6

Step 2

batch	i=3,4	i=1,2	i=5,6

Step 3

batch	i=3,4	i=1,2	i=5,6
T/W	5/4	5/2	6/2
sequence	first	second	third

Step 4

▨	Processing in the batch machine
□	Processing on the unary machine

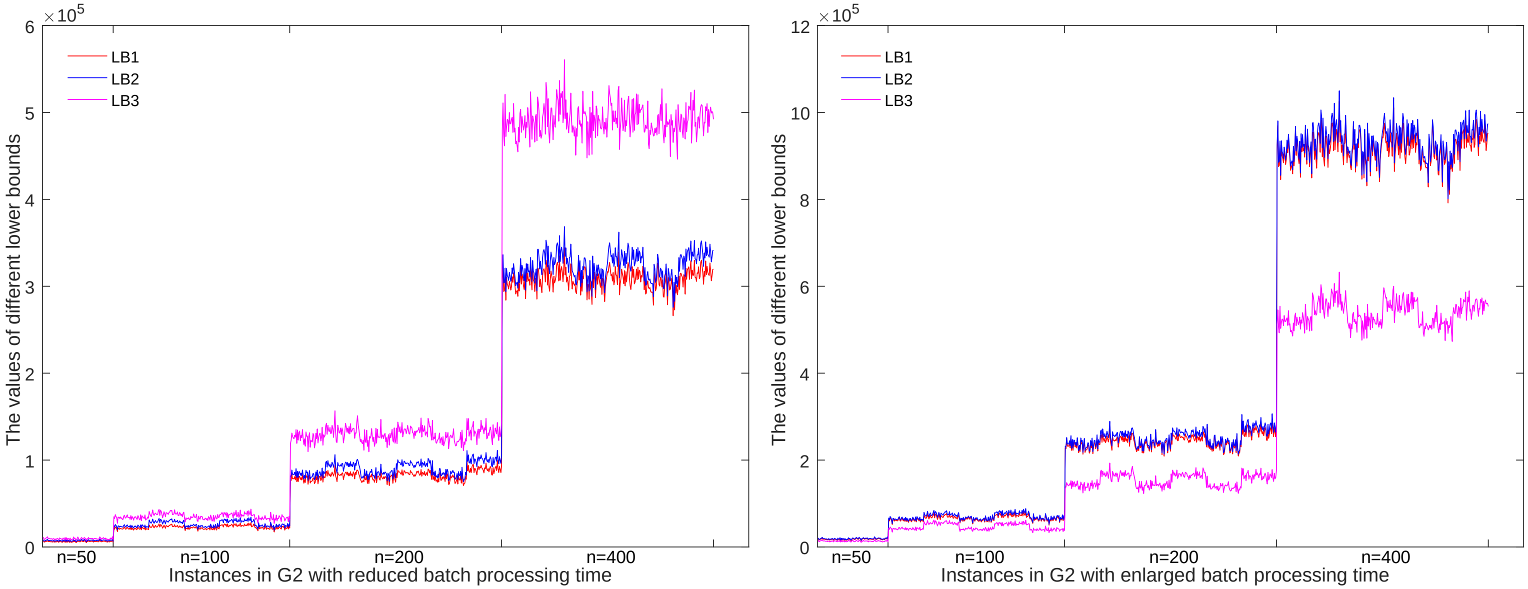
- The numbers in the Gantt chart represent the job indices.
- During the processing of job 3, the second batch has been completed in the batch machine. Since job 1 has smaller ratio than job 3 ($p_1/w_1<p_3/w_3$), job 3 is preempted by job 1 on the unary machine.

Step 5: Convertor I

Step 6: Convertor II

Objective value: 115

Objective value: 105 (**Step 7**: the solution of the GBDS algorithm)


$\times 10^5$
6
5
4
3
2
1
0
LB1
LB2
LB3
The values of different lower bounds
n=50
n=100
n=200
n=400
Instances in G2 with reduced batch processing time
$\times 10^5$
12
10
8
6
4
2
0
LB1
LB2
LB3
The values of different lower bounds
n=50
n=100
n=200
n=400
Instances in G2 with enlarged batch processing time