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

Journal:	*International Journal of Production Research*
Manuscript ID	TPRS-2017-IJPR-1071
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	03-Jun-2017
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

Submitted to *International Journal of Production Research*

Minimizing Total Weighted Completion Time on Batch and Discrete 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, 1–19

Minimizing Total Weighted Completion Time on Batch and Discrete 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 discrete 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. Numerical results demonstrate that the proposed algorithm can obtain high quality solutions and have a competitive performance. Sensitivity analysis indicates that the performance of the proposed algorithm is also robust on different problem structures.

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

1. Introduction

This paper focuses on a two-stage scheduling problem comprised of a batch machine in the upstream and a discrete 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 discrete 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 discrete machine. This hybrid process with a batch machine and a discrete 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 discrete 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 discrete machines.

We denote this two-stage scheduling problem as $\beta \rightarrow \delta|incompat|\sum w_j C_j$, where β, δ and *incompat* mean the batch machine, discrete machine and incompatible job families, respectively.

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

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 $\beta \to \delta|\sum C_j$. 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 (2016) and Huang et al. (2017) studied the two-stage scheduling problem with limited waiting time to minimize C_{max}. The released 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.

To our knowledge, the problem $\beta \to \delta|incompat|\sum w_j C_j$ has not been studied in the literature. The goal of this research is to develop a constant approximation algorithm for this problem to fill this gap. In this paper, we first propose a mixed-integer linear programming model for problem $\beta \to \delta|incompat|\sum w_j C_j$. 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. Numerical results show that the proposed algorithm has competitive and robust performances compared with the meta-heuristics.

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. Computational experiments are conducted in Section 5, and we conclude this paper in Section 6.

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. In the second stage, the jobs are processed one by one in an arbitrary sequence on the discrete machine. All the jobs are available at time zero. We assume that the number of jobs of each family is an integer multiple of the batch capacity. Preemption is not allowed on both the batch machine and discrete machine. For a better presentation, the notations and decision variables are illustrated as follows.

Notations

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 discrete machine;
w_i : the weight of job i.
q_k : the processing time of batches consisting 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 discrete machine, and otherwise, $u_{ij} = 0$;
C_i : the completion time of job i on the discrete 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* $\beta \rightarrow \delta|incompat|\sum w_j C_j$, *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 discrete machine unchange. 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 discrete 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 above notations and decision variables, this problem can be formulated as follows.

$$(BDMI)\ \min \sum_{j\in N} w_j C_j \tag{1}$$

$$\text{s.t.} \sum_{l\in B} x_{il} = 1, \qquad \forall i \in N, \tag{2}$$

$$\sum_{i\in N} x_{il} = b, \qquad \forall l \in B, \tag{3}$$

$$\sum_{k\in F} y_{kl} = 1, \qquad \forall l \in B, \tag{4}$$

$$\sum_{l\in B} y_{kl} = \frac{n_k}{b}, \qquad \forall k \in F, \tag{5}$$

$$x_{il} \le y_{kl}, \qquad \forall i \in F_k, l \in B, \tag{6}$$

$$u_{ij} + u_{ji} = 1, \qquad \forall i, j \in N, i < j, \tag{7}$$

$$C_i - p_i \ge C_j - M_1 u_{ij}, \qquad \forall i, j \in N, i \ne j, \tag{8}$$

$$C_i - p_i \ge \sum_{h=1}^{l} \sum_{k\in F} y_{kh} q_k - M_2(1 - x_{il}), \qquad \forall i \in N, l \in B \tag{9}$$

$$u_{ij} \in \{0, 1\}, \qquad \forall i, j \in N, i \ne j, \tag{10}$$

$$y_{kl} \in \{0, 1\}, \qquad \forall k \in F, l \in B, \tag{11}$$

$$x_{il} \in \{0, 1\}, \qquad \forall i \in N, l \in B. \tag{12}$$

The objective function (1) is to minimize the total weighted completion time. Constraint (2) make 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) makes sure that jobs in family k should be assigned to n_k/b batches. Constraint (6) shows that only when batch l consists of the jobs in family k, the jobs in family k can be assigned to batch l. Constraint (7) makes sure that jobs have processing precedence on the discrete machine. Constraint (8) makes sure that only when a job finishes processing on the discrete machine, the latter ones can start processing on the discrete machine. Constraint (9) makes sure that job i can be processed on the discrete machine only when the batch that job i belongs to finishes processing on the batch machine. Constraints (10), (11) and (12) are the range of the variables. Herein, we can set $M_1 = \sum_{k\in F} \frac{n_k q_k}{b} + \sum_{i\in N} p_i$ and $M_2 = \sum_{k\in F} \frac{n_k q_k}{b}$.

If we set the number of incompatible job families and all the weights to be 1, this problem reduces to problem $\beta \to \delta|\sum C_j$, which has been shown to be strongly NP-hard by Ahmadi et al. (1992). Thus the problem $\beta \to \delta|incompat|\sum w_j C_j$ is also NP-hard in strong sense. We use the formulation BDMI customized for the full-batch data to conduct the computational experiments.

3. Lower bounds

In this section, we develop three lower bounds for problem $\beta \to \delta|incompat|\sum w_j C_j$, 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 decreasing 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, there exists an optimal schedule, where all batches are sequenced in increasing 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 discrete machine immediately, i.e., the waiting time on the discrete 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*

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

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. 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 buffer and the processing time p_j^* on the discrete 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 $j \in N$. Let $\hat{p}_j$ and $\hat{w}_j$ denote the related processing time and weight of job j after sorting and define Γ_{WSPT} as the total weighted completion time obtained by the weighted shortest processing time (WSPT) rule for the single discrete machine, i.e., $\Gamma_{\mathrm{WSPT}} = \sum_{i \in N} \sum_{j \leq i} \hat{w}_i \hat{p}_j$.*

The WSPT rule gives the optimal solution for the single discrete machine to minimize the total weighted completion time. In Eastman, Even, and Isaacs (1964), the authors proposed the lower bound for the problem of scheduling n jobs on some identical discrete 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 discrete machines with the objective to minimize the total weighted completion time.*

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

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

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 discrete 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

$$\begin{aligned}\sum_{j=1}^{n} w_j^* C_j^* &\geq \sum_{j=1}^{n} w_j^* C_j^0 \\ &= \sum_{l=1}^{a}\sum_{i=1}^{b} WE_{li}^* CT_{li}^* + \sum_{l=1}^{a}\sum_{i=1}^{b} WE_{li}^* \sum_{j\leq i} PT_{lj}^* \\ &\geq \Gamma_{GRWC} + \sum_{l=1}^{a}\sum_{i=1}^{b} WE_{li}^* \sum_{j\leq i} PT_{lj}^*,\end{aligned}$$

where CT_{lj}^*, PT_{lj}^* and WE_{lj}^* mean the completion time on the batch machine, the processing time and the weight of job j assigned to batch l in the optimal solution, respectively. The second term of the last expression is same with the problem of scheduling n jobs on a identical discrete machines with additional constraints that each discrete machine must process b jobs. According to Lemma 3, $\frac{1}{a}\Gamma_{WSPT} + \frac{a-1}{2a}\sum_{i\in N} w_i p_i$ is smaller than or equal to the second term. Thus we have

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

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 discrete machine after the first batch is finished at time r_1, this problem is relaxed to the single discrete 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 of 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 discrete 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 enter the buffer earlier. On the other hand, if the batch processing time is large enough, the jobs within one batch should be processed on the discrete 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. Based on the preemptive schedule, a 4-approximation algorithm is proposed to solve this problem. Each algorithm consists 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 discrete machine. We denote the approximation algorithm as GBDS (greedy batch 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 discrete machine is busy. It is described as follows.

Preemptive GBDS

Step 1. For each k, sort jobs of F_k in the decreasing 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. The batches are processed on the batch machine in increasing order of T_l/W_l;
Step 4. When the discrete machine is available, select the job i^* which has the smallest value of p/w in the current buffer,

- If no batches are finished during the processing of job i^*, process job i^*;
- Else, next batch l^* is finished and enters the buffer;

- If the ratio p_i^*/w_i^* is smaller than or equal to that of all jobs in the batch l^*, process job i^*;
- Else, process job i^* partially until the finished time of batch l^*. Then stop processing job i^* and return the remaining part of job i^* with the criterion p_i^*/w_i^* to the buffer;

Step 5. Repeat the process in Step 4 until all jobs are completed. Return the schedule π.

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: $p_{l1}/w_{l1} \leq p_{l2}/w_{l2} \leq p_{l3}/w_{l3}$ for batch $l = 1, 2, 3, 4$; $p_{23}/w_{23} \leq p_{31}/w_{31}$; $p_{42}/w_{42} \leq p_{32}/w_{32}$ and $p_{33}/w_{33} \leq p_{43}/w_{43}$. 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.

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*

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

Proof. Let C_j^P, t_j^P, Δ_j^P, p_j^P, w_j^P denote the completion time on the discrete machine, the finished time on the batch machine, the waiting time in the buffer, the processing time on the discrete machine and the weight for job j in the solution of the preemptive GBDS algorithm, respectively. For each job j, we have

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

For a preempted job, the completion time of this job is equal to the completion time of the last part of this job. Note that there is no idle time between t_j^P and C_j^P for any job j. When one batch is finished and the discrete machine is busy, there are two cases:

- Case 1. The preemption occurs, such as the batch 4 in Figure 1. For job j in this batch, the waiting time Δ_j^P is the sum of the processing time of jobs of which ratios p/w are smaller than $\frac{p_j^P}{w_j^P}$ in the buffer. It is smaller than or equal to the sum of the processing time of jobs of which ratios p/w are smaller than $\frac{p_j^P}{w_j^P}$ in N, i.e., $\sum_{i<\hat{j}} \hat{p}_i$, where $\hat{j}$ is the position of job j in the WSPT order;
- Case 2. There is no preemption, such as the batch 3 in Figure 1. For job j in this batch, the waiting time Δ_j^P is the sum of the processing time of jobs of which ratios p/w are smaller than $\frac{p_j^P}{w_j^P}$ in the buffer and partial processing time of the job i_0 which occupies the discrete machine when the batch is finished, such as job 23 in Figure 1. But because the ratio $\frac{p_{i_0}}{w_{i_0}}$ is smaller than $\frac{p_j^P}{w_j^P}$, Δ_j^P is also smaller than $\sum_{i<\hat{j}} \hat{p}_i$ which contains p_{i_0}.

When one batch is finished and the discrete machine is available, it is easy to check that the waiting time Δ_j^P is the sum of the processing time of which $\frac{p_i^P}{w_i^P}$ is smaller than $\frac{p_j^P}{w_j^P}$ in the buffer. It is smaller than or equal to $\sum_{i<\hat{j}} \hat{p}_i$, where $\hat{j}$ is the position of job j in the WSPT order. Therefore

we have the following inequalities

$$\begin{aligned} Z(\pi_P) &= \sum_{j\in N} w_j^P (r_j^P + \Delta_j^P + p_j^P) \\ &\le \sum_{j\in N} w_j^P (r_j^P + \sum_{i<\hat{j}} \hat{p}_i + p_j^P) \\ &= \sum_{j\in N} w_j^P r_j^P + \sum_{j\in N} w_j^P \sum_{i\le\hat{j}} \hat{p}_i \\ &= \Gamma_{GRWC} + \Gamma_{WSPT} \end{aligned}$$

This completes the proof. □

Figure 1. An illustration of the preemptive GBDS algorithm.

4.2 *GBDS algorithm*

The solution of the preemptive GBDS algorithm is not feasible for the problem $\beta \rightarrow \delta|incompat|\sum w_jC_j$, since preemption exists. Based on the preemptive solution, we derive the GBDS algorithm to obtain a feasible solution as follows.

GBDS

Step 1. For each k, sort jobs of F_k in the decreasing 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. The batches are processed on the batch machine in increasing order of T_l/W_l;
Step 4. Run the preemptive GBDS algorithm. Form a list of the jobs, ordered by their preemptive completion times C_j^P on the discrete machine;
Step 5. Adjust the schedule non-preemptively using the list, with the constraint that no job j starts before its finished time t_j on the batch machine,

- *Convertor I.* Consider the last scheduled piece of any job j, which has length k_j, and is scheduled from time $C_j^P - k_j$ to C_j^P in the preemptive schedule. Insert $p_j - k_j$ extra units of time in the schedule at time C_j^P. Schedule job j non-preemptively in the resulting available block of length p_j. Denote the schedule as π_1;
- *Convertor II.* Consider the first scheduled piece of any job j, which has length k_j, and is scheduled from time $C_j^P - k_j$ to C_j^P in the preemptive schedule. Insert $p_j - k_j$ extra units of time in the schedule at time C_j^P. Schedule job j non-preemptively in the resulting available block of length p_j. Denote the schedule as π_2;

Step 6. Identify the index: $s^* = \arg\min_s Obj(\pi_s)$, $s = 1, 2$. Return the schedule π_{s^*}.

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*

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

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 released time to the discrete machine. So the sequence decisions on the discrete 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 could produce 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

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

This completes the proof. □

5. Computational experiments

This section conducts 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 BDMI is solved by Cplex 12.6 with a time limit 3600 seconds under default configuration.

5.1 *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 discrete 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 discrete 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 BDMI. 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.

5.2 *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 BDMI formulation;
Time: the average time in seconds to solve the instances to optimality by the BDMI formulation;
Gap_1: the average gap between GBDS and optimal solutions;
Gap_2: the average gap between LB and optimal solutions.

In Table 1, the columns of *Time*, Gap_1 and Gap_2 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 BDMI formulation, the *#Opt* is zero and the columns of *Time*, Gap_1 and Gap_2 are displayed as "-". As shown in Table 1, the problem size that can be solved to optimality by BDMI 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_1 are all very small. The maximum average value of Gap_1 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_2 appearing in the combination 8-4-2 is 4.81%.

5.3 *Comparison results for instances in* G_2

For instances in G_2, we use the lower bound and two population-based meta-heuristics, i.e., Differential Evolution (DE) and Harmony Search (HS), to test the efficiency of the proposed algorithm.

DE is first proposed by Storn and Price (1997), which has been widely applied to various optimisation problems recently (Fu, Sivakumar, and Li 2012; Zhang et al. 2017). The mutation operator is as below,

$$V_i(g+1) = X_\alpha(g) + \lambda(best(g) - X_\alpha(g)) + F(X_\beta(g) - X_\gamma(g)).$$

where $X_i(g)$ is the ith individual at generation g and $V_i(g+1)$ is a mutant individual for target individual $X_i(g)$. This mutation operator is proved to have good convergent performance in Fu, Sivakumar, and Li (2012). The crossover operator, selection operator and parameter settings of DE are referred to Fu, Sivakumar, and Li (2012). In detail, the values of three control variables are that $\lambda = 0.4$, $F = 0.7$, $CR = 0.5$. The population size is set as $2a$.

HS, mimicking the improvisation process of music players, is recently developed by Geem, Kim, and Loganathan (2001). HS has also been very successful in a wide variety of optimization problems (Mahdavi, Fesanghary, and Damangir 2007; Zou et al. 2010; Guo et al. 2017). The parameter

settings are referred to Mahdavi, Fesanghary, and Damangir (2007). The harmony memory considering rate (HMCR) is set as 0.95. The pitch adjusting rate (PAR) is set as 0.6. The harmony memory size (HMS) is equal to 5. The band width (bw) is equal to 0.2.

The time limit for DE and HS is set as 20 minutes. To apply DE and HS to solve this problem, we need to code the feasible solutions. The code consists of $m+1$ parts. The former m parts are used to decide the batch forms in m job families. The last part is used to get the batch sequences on the batch machine. The job sequences on the discrete machine is same as that in the batch machine, i.e., follows first in and first out rule. The value of each position in the code is randomly generated from uniform distribution $[-1, 1]$. The related sequence is based on the decreasing order of the code value. We give an example in which $n = 8, m = 2, b = 2$ to illustrate the code process in Figure 2.

Figure 2. Illustration on the code process of HS and DE.

Figure 3 graphically describes the average relative gaps generated by GBDS, HS and DE with different batch capacities (b = 10 and b = 20) compared to the lower bounds for instances in G_2. As shown in Figure 3, the average relative gaps obtained by the GBDS are the lowest in most combinations except the combination 50-5-10. Moreover, with the problem size increasing, we find that the relative gaps of the GBDS tend to be stable. However, the relative gaps of the HS and DE fluctuate greatly. When the batch forms in each family are simple, HS and DE tend to have good performances, such as the combinations 50-5-10, 100-10-10, 100-5-20 and 200-10-20, in which each family only has one batch.

Figure 3. Comparison results of GBDS, HS and DE on the average gap for different buffer sizes.

Table 2 presents the detail comparison results for instances in G_2. *Gap-GBDS*, *Gap-HS* and *Gap-DE* 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, HS or DE). *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 time seconds for GBDS are also very small. These results indicate that the GBDS algorithm can obtain high quality solutions in a short time. Most values of *Gap-GBDS* are less than those of *Gap-HS* and *Gap-DE*, which shows that the GBDS algorithm seems to yield better results than the other two population-based algorithms.

To further justify this observation, we establish the null hypothesis as H_0: *Gap-GBDS*$-$*Gap-HS*≥ 0 and the alternative hypothesis H_1: *Gap-GBDS*$-$*Gap-HS*< 0 when GBDS compares to HS. When GBDS compares to DE, the hypothesises are H_0: *Gap-GBDS*$-$*Gap-DE*≥ 0 and H_1: *Gap-GBDS*$-$*Gap-DE*< 0. In Table 2, P_1 and P_2 represent the p-values obtained by the T-test when GBDS compares to HS and DE, respectively. The null hypothesises of most combinations are rejected when compared to HS , since the p-values are less than the 5% significance level except the combination 50-5-10. These results show that the proposed GBDS algorithm has a competitive performance when compared to the HS algorithm. When compared to the DE algorithm, the null hypothesises are rejected in all the combinations, which demonstrates that the proposed GBDS algorithm can find better solutions than the DE algorithm.

5.4 *Sensitivity analysis*

The relationship between the batch processing time and the sum of the processing time of jobs in one batch on the discrete 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 family k, respectively. The results are reported in Table 3 and Table 4. The columns are same with that in Table 2. The parameter settings and termination conditions of HS and DE are kept same with the subsection 5.3.

When the batch processing time is reduced, the batches are finished earlier and more batches enter the buffer to wait to be processed on the discrete 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. The hypothesis test results show that the proposed algorithm outperforms the HS and DE algorithms.

When the batch processing time is enlarged, the jobs in one batch tend to be processed on the discrete 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 compared to the HS and DE algorithms, the p-values are all less than 5%, which indicates that the proposed algorithm still has a competitive performance when enlarging the batch processing time.

The above results show that GBDS can obtain high quality solutions and has a competitive performance in different problem structures. The results are very robust, which shows that the GBDS algorithm has great potential for solving the practical two-stage batch scheduling problem.

6. Conclusions

In this paper, we study the two-stage scheduling problem with a batch machine followed by a discrete 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 a constant 4. Numerical experiments are conducted to test the efficiency of the proposed algorithm. When compared to the optimal solutions and lower bounds, the proposed algorithm can obtain high quality solutions. When compared to the population-based heuristics, Harmony Search and Differential Evolution, the proposed algorithm has a competitive performance. Sensitivity analysis indicates that the proposed algorithm also has 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 approximation algorithm for this problem should be explored in the future.

Acknowledgements

This research was supported in part by National Science Foundation of China [grant numbers 71690232 and 71371015] and by National Science Foundation [grant numbers CMMI-1435800 and CMMI-1536978].

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. 2016. "Two new approaches for a two-stage hybrid flowshop problem with a single batch processing machine under waiting time constraint." *Computers & Industrial Engineering*, http://dx.doi.org/10.1016/j.cie.2016.11.031.

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.

Geem, Z. W., J. H. Kim, and G. V. Loganathan. 2001. "A New Heuristic Optimization Algorithm: Harmony Search." *Simulation* 76 (2): 60–68.

Guo, Z., L. Shi, L. Chen, and Y. Liang. 2017. "A harmony search-based memetic optimization model for integrated production and transportation scheduling in MTO manufacturing." *Omega* 66: 327–343.

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*, http://dx.doi.org/10.1016/j.cie.2017.04.010.

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

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.

Mahdavi, M., M. Fesanghary, and E. Damangir. 2007. "An improved harmony search algorithm for solving optimization problems." *Applied Mathematics and Computation* 188 (2): 1567–1579.

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.

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

Storn, R., and K. V. Price. 1997. "Differential Evolution-A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces." *Journal of Global Optimization* 11 (4): 341–359.

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.

Zou, D., L. Gao, J. Wu, S. Li, and Y. Li. 2010. "A novel global harmony search algorithm for reliability problems." *Computers & Industrial Engineering* 58 (2): 307–316.

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

n-m-b	*#Opt*	*Time (s)*	*Gap*$_1$ *(%)*	*Gap*$_2$ *(%)*
8-2-2	10	0.17	2.16	4.55
8-2-4	10	0.08	0.59	1.82
8-4-2	10	0.10	2.15	4.81
12-2-2	10	115.14	3.49	4.06
12-2-4	10	9.45	0.46	2.56
12-4-2	10	3.64	2.99	3.66
16-2-2	1	987.78	0.60	1.42
16-2-4	9	1137.82	1.68	2.24
16-4-2	3	2626.43	2.11	2.80
16-4-4	10	22.13	1.73	2.24
20-2-2	0	-	-	-
20-2-4	0	-	-	-
20-4-2	0	-	-	-
20-4-4	4	2484.88	1.54	1.60

Table 2. Comparison results of GBDS, HS and DE on the instances in G_2.

n-m-b	*Gap-GBDS (%)*	*Time (s)*	*Gap-HS (%)*	*Gap-DE (%)*	P_1	P_2
50-2-10	2.44	0.27	4.74	6.73	1.68E-16	1.39E-25
50-5-10	2.12	0.26	1.77	2.68	9.99E-01	1.28E-05
100-2-10	2.44	0.52	7.69	9.64	3.94E-34	2.78E-43
100-2-20	1.46	0.50	5.92	8.22	8.27E-36	3.04E-43
100-5-10	2.26	0.52	5.52	7.51	4.00E-31	1.85E-41
100-5-20	1.25	0.50	1.71	2.76	5.32E-10	2.36E-26
100-10-10	3.10	0.52	3.24	4.02	3.07E-02	2.23E-14
200-2-10	2.18	1.18	10.03	11.37	1.88E-50	6.03E-53
200-2-20	1.60	1.12	8.56	10.43	2.76E-50	6.21E-54
200-5-10	2.07	1.17	8.24	10.14	6.07E-45	7.71E-50
200-5-20	1.59	1.12	6.24	8.42	8.72E-40	1.67E-50
200-10-10	2.52	1.18	6.67	8.60	5.49E-39	4.56E-41
200-10-20	1.37	1.12	2.24	2.89	1.12E-25	4.77E-33
400-2-10	1.80	2.77	11.44	11.98	3.22E-58	8.23E-61
400-2-20	1.39	2.65	10.58	11.53	3.82E-59	4.19E-58
400-5-10	1.83	2.83	10.30	11.52	3.43E-57	3.39E-58
400-5-20	1.36	2.56	9.27	10.71	6.52E-57	1.08E-59
400-10-10	2.73	2.82	9.48	11.64	8.19E-49	1.02E-51
400-10-20	1.47	2.76	7.43	9.03	3.00E-50	5.00E-58

Note: The time limit for DE and HS is set as 20 minutes.

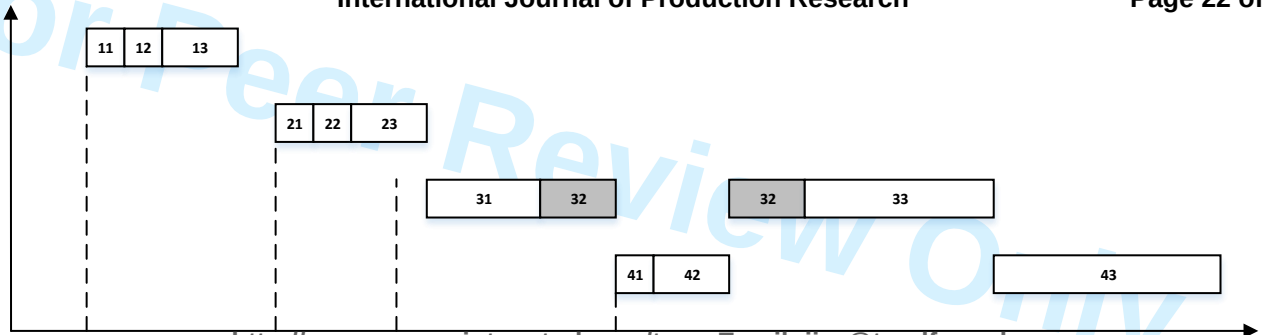
Table 3. Comparison results for instances of G_2 when reducing the batch processing time.

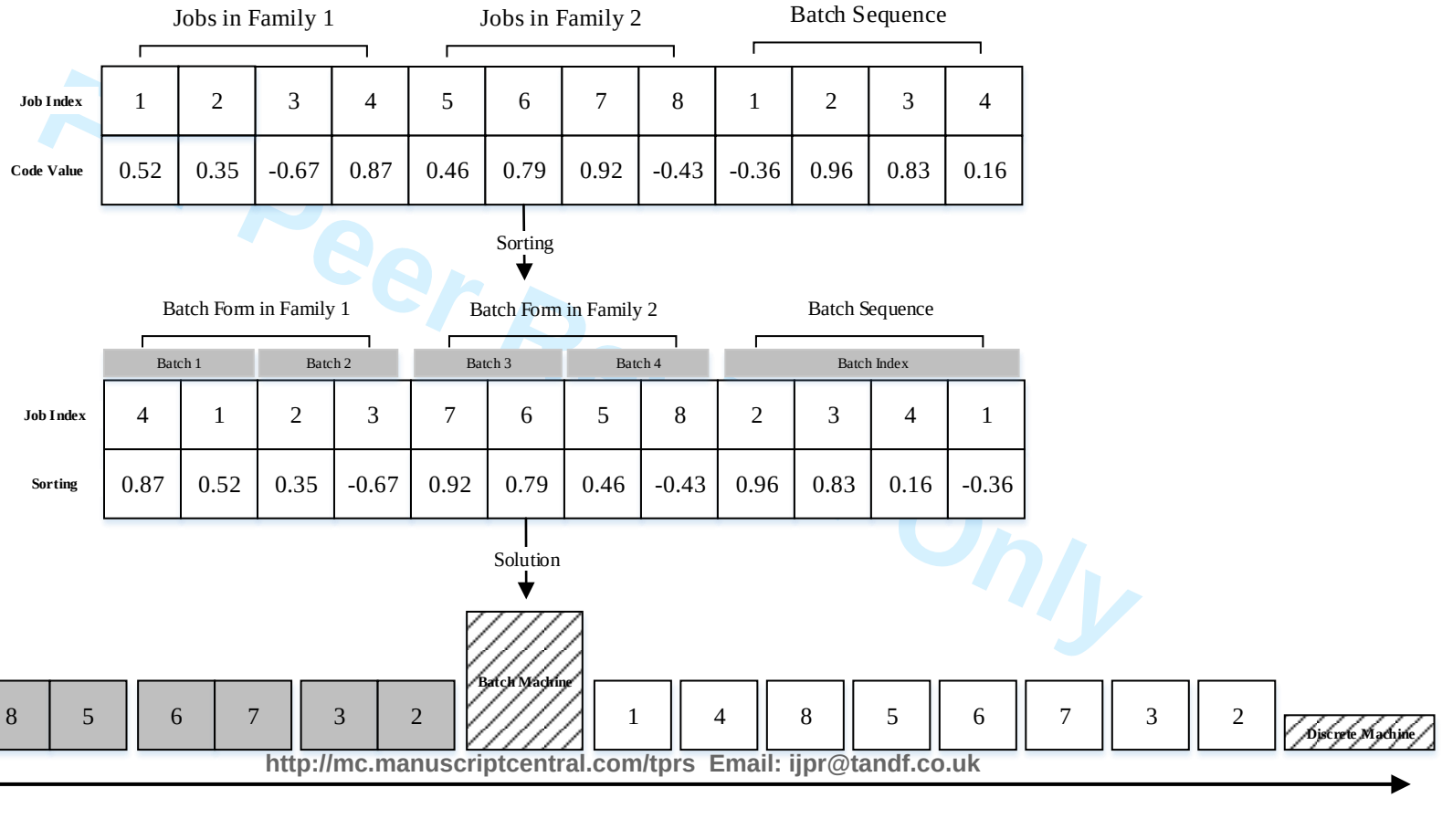
n-m-b	*Gap-GBDS (%)*	*Gap-HS (%)*	*Gap-DE (%)*	P_1	P_2
50-2-10	5.59	6.34	11.82	2.93E-03	1.39E-21
50-5-10	7.98	19.06	20.21	1.74E-32	1.58E-34
100-2-10	6.12	13.09	20.06	4.70E-23	5.06E-41
100-2-20	4.75	9.19	15.95	7.05E-22	7.13E-34
100-5-10	7.19	11.59	18.97	1.26E-21	5.39E-36
100-5-20	7.66	21.77	23.35	4.16E-41	3.48E-42
100-10-10	9.66	24.76	25.92	2.46E-39	1.57E-40
200-2-10	6.97	22.14	27.45	1.13E-43	1.14E-50
200-2-20	6.10	17.72	24.77	6.85E-39	2.09E-47
200-5-10	7.07	19.30	26.54	2.42E-39	7.06E-48
200-5-20	6.89	15.74	23.26	1.72E-43	5.91E-48
200-10-10	7.86	18.07	26.33	1.93E-37	1.17E-47
200-10-20	9.71	28.14	29.30	7.95E-50	8.93E-50
400-2-10	7.42	30.62	32.75	5.06E-58	9.30E-59
400-2-20	6.88	27.02	31.11	6.43E-50	8.57E-60
400-5-10	7.31	27.21	31.55	2.60E-51	1.68E-58
400-5-20	6.90	23.46	29.86	4.10E-49	9.99E-55
400-10-10	7.49	24.98	31.35	6.34E-51	6.33E-62
400-10-20	7.80	22.81	30.09	3.15E-56	2.49E-61

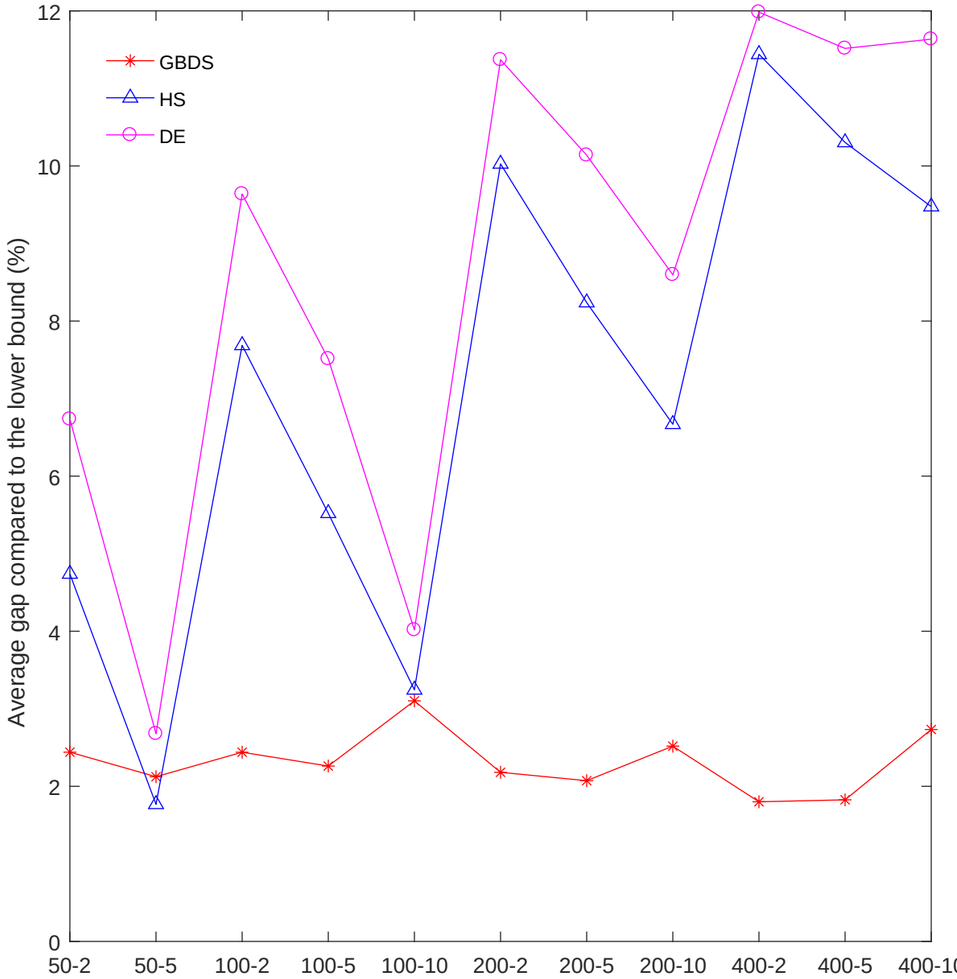
Table 4. Comparison results for instances of G_2 when enlarging the batch processing time.

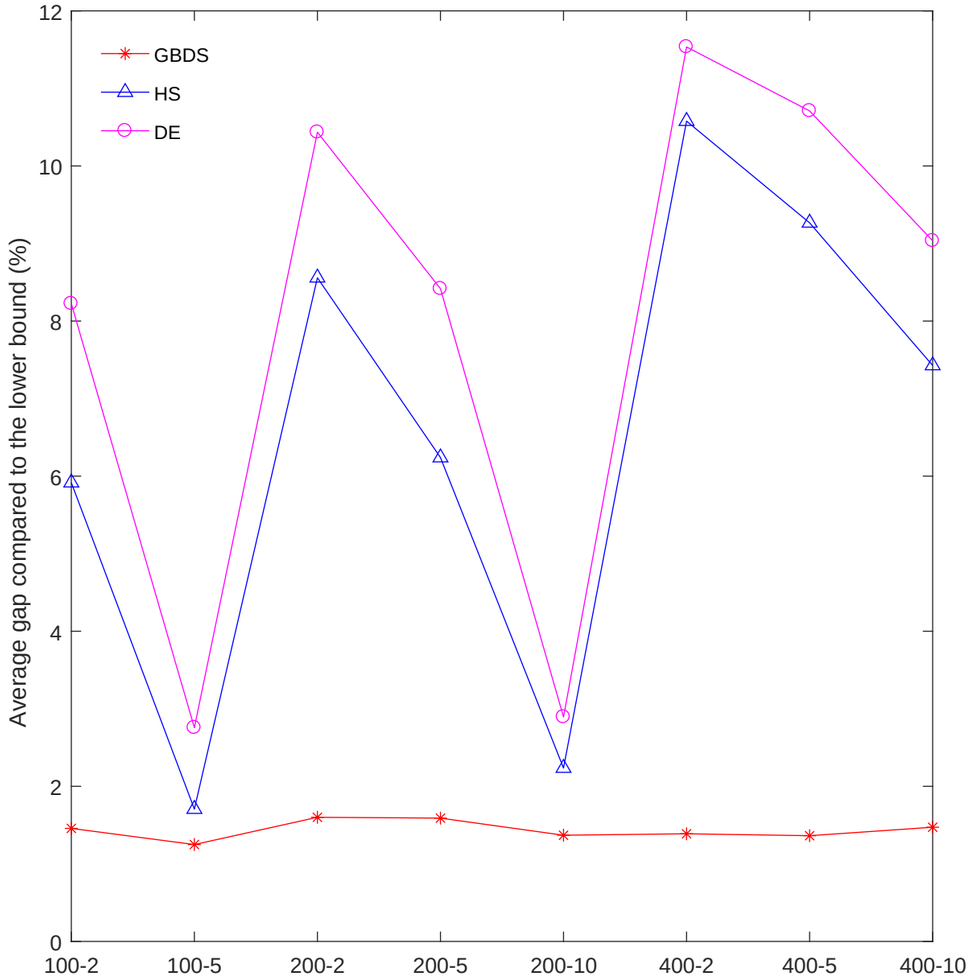
n-m-b	*Gap-GBDS (%)*	*Gap-HS (%)*	*Gap-DE (%)*	P_1	P_2
50-2-10	0.44	2.90	4.65	3.74E-32	1.23E-38
50-5-10	0.25	0.39	0.91	1.06E-30	1.93E-27
100-2-10	0.30	5.14	7.22	1.04E-42	6.28E-52
100-2-20	0.33	4.35	6.59	5.11E-40	3.70E-49
100-5-10	0.26	3.19	5.23	5.35E-41	7.64E-46
100-5-20	0.12	0.54	1.26	5.98E-40	4.96E-41
100-10-10	0.17	0.61	1.20	1.90E-36	2.24E-37
200-2-10	0.17	7.53	8.89	6.67E-54	1.66E-60
200-2-20	0.22	6.58	8.45	3.29E-48	4.66E-63
200-5-10	0.16	5.90	7.83	4.11E-54	1.75E-60
200-5-20	0.19	4.21	6.50	2.32E-50	9.41E-56
200-10-10	0.15	4.01	5.70	2.87E-48	1.60E-52
200-10-20	0.08	0.71	1.26	7.75E-45	1.08E-49
400-2-10	0.09	9.28	10.03	8.56E-65	1.24E-70
400-2-20	0.12	8.65	9.93	3.90E-61	5.69E-72
400-5-10	0.09	8.20	9.49	7.15E-59	1.06E-63
400-5-20	0.12	7.25	9.11	2.13E-61	3.96E-72
400-10-10	0.09	6.32	8.01	2.55E-60	5.98E-61
400-10-20	0.10	5.39	7.17	2.47E-60	2.44E-67

Figure 1. An illustration of the preemptive GBDS algorithm.
Figure 2. Illustration on the code process of HS and DE.
Figure 3. Comparison results of GBDS, HS and DE on the average gap for different buffer sizes.






GBDS
HS
DE
Average gap compared to the lower bound (%)
0
2
4
6
8
10
12
50-2
50-5
100-2
100-5
100-10
200-2
200-5
200-10
400-2
400-5
400-10
Combination of n and m when b=10


GBDS
HS
DE
Average gap compared to the lower bound (%)
0
2
4
6
8
10
12
100-2
100-5
200-2
200-5
200-10
400-2
400-5
400-10
Combination of n and m when b=20