Front. Eng. Manag. 2017, 4(3): 368–374
DOI 10.15302/J-FEM-2017034

RESEARCH ARTICLE

**Zhongshun SHI, Zewen HUANG, Leyuan SHI**

# Two-stage scheduling on batch and single machines with limited waiting time constraint



**Abstract** This study addresses the problem of two-stage scheduling on batch and single machines with limited waiting time constraint; thus, the makespan is minimized. A mixed-integer linear programming model is proposed for this problem. Three tight lower bounds and a heuristic algorithm are developed. The worst-case performance of the proposed algorithm is discussed. A hybrid differential evolution algorithm is also developed to improve the solution quantity. Numerical results show that the hybrid algorithm is capable of obtaining high-quality solutions and exhibits a competitive performance

**Keywords** batch machine, flow shop, makespan, limited waiting time

## 1 Introduction

This study focuses on a hybrid two-stage flow shop scheduling problem. In the first stage, all of the jobs must be processed on a batch machine. The batch machine can simultaneously process several jobs up to its capacity. In the second stage, a single machine handles these jobs individually. In the two stages, the waiting time that elapses between the completion time of the first stage and the start time of the second stage is limited.

This problem is prompted by the heat treatment process in a real rocket assembly plant. The sheet metals must be heated in an oven to eliminate stress-corrosion to meet the requirements of rocket-driven spacecraft. These sheet metals are moved to a single machine for further trimming after heat treatment. The sheet metals must be processed within 2 h; otherwise, the trimming operation cannot be conducted because the sheet metals become hard, thereby leading to waste of materials. The heat treatment operation is a bottleneck process. On one hand, the heat treatment process is time-consuming. On the other hand, forming batches and the sequence that should be followed mainly depend on the manual experience. The two factors cause unbalanced production and low efficiency.

Batch machines can find their widespread applications in various industrial systems, such as heat treatment ovens in the steel industry, burn-in operation in the semiconductor final test, and oxidation and diffusion in wafer fabrication. The two-stage scheduling problem on batch and single machines has attracted considerable interest. Ahmadi et al. (1992) first studied the problem and proposed a full batch–longest processing time (LPT) rule in order to minimize the makespan and a full batch dealing–shortest processing time heuristic to minimize the total completion time. Consequently, Hoogeveen and Velde (1998) considered the objective of minimizing total completion time, and proposed a Lagrangian lower bound whose time complexity is O(*nlogn*) based on the concept of positional completion time. Gong et al. (2010) considered the two-stage scheduling problem to minimize the sum of the makespan and total blocking time. The batch remains in the batch machine if the single machine was busy after completing one batch. Several approximation algorithms were proposed. Zhang et al. (2017) analyzed the two-stage batch scheduling problem with incompatible job families and a limited buffer. Approximation and hybrid differential evolution algorithms were proposed.

Received May 1, 2017; accepted August 26, 2017

Zhongshun SHI
Department of Industrial & Systems Engineering, University of Wisconsin-Madison, WI Madison 53706, USA

Zewen HUANG
Department of Industrial Engineering & Management, Peking University, Beijing 100871, China

Leyuan SHI (✉)
Department of Industrial & Systems Engineering, University of Wisconsin-Madison, WI Madison 53706, USA; Department of Industrial Engineering & Management, Peking University, Beijing 100871, China
E-mail: leyuan@coe.pku.edu.cn

All of the reviewed research ignored the limited waiting time constraint. Only a few studies addressed the problems related to the two-stage scheduling with the batch machine and limited waiting time constraint. Su (2003) considered the nonidentical limited waiting time constraint for the two-stage scheduling problem, and proposed a mixed-integer linear model and a heuristic algorithm to minimize the makespan. However, the number of jobs in the computational experiments is minimal. The lower bound proposed by Su (2003) is calculated by relaxing the limited waiting time constraint. In the current study, we investigate the two-stage scheduling problem with identical limited waiting time. The problem size is extended in the experiments, and we derive three fast-computing lower bounds. Chung et al. (2016) examined the two-stage scheduling problem with limited waiting time constraint and considered the release time and nonidentical job sizes. However, in the study of Huang et al. (2017), the mathematical programming model was revised and several properties of this kind of problem were identified.

In the current study, a mixed-integer linear programming model is proposed to describe the problem. Three lower bounds and a heuristic algorithm, together with the worst-case analysis, are proposed. The hybrid differential evolution algorithm (HDE) algorithm integrating the solution of the heuristic algorithm is developed to improve the solution quantity. The numerical experiments are conducted to test the performance of the hybrid algorithm.

The remainder of this paper is organized as follows: Section 2 presents the problem definition and mathematical formulation. Section 3 discusses the development of three lower bounds for this problem and presents a heuristic algorithm and the worst-case analysis. Section 4 describes the proposed HDE algorithm. Section 5 presents the numerical experiments. Section 6 concludes this paper.

## 2 Problem description and formulation

We first provide a formal description of the problem. $n$ jobs will be processed on one batch machine and one single machine. In the first stage, these jobs are divided into several batches to be processed on the batch machine. In the second stage, the jobs are processed individually in an arbitrary sequence on the single machine. All of the jobs are available at the beginning of the time horizon and have identical size. Each batch machine can handle several jobs simultaneously up to its capacity $b$. The waiting time, the time that elapses between the completion time of the first stage and the start time of the second stage, is limited by $w$ for each job. The processing time of each batch is $t$, which is independent of the jobs. The processing time of job $i$ on the single machine is $p_i$, $i = 1,\ldots,n$. Without loss of generality, we sort the jobs in nonincreasing order of $p_i$, where $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_n$. Preemption is restricted on the batch and single machines. The objective is to determine the batch forms, batch sequence, and job sequence to minimize the makespan.

Before formulating the problem, we introduce a property in Lemma 1, which was proven by Su (2003).

**Lemma 1.** (Su, 2003) An optimal schedule exists, in which the jobs in each batch are consecutively sequenced at the second stage.

According to Lemma 1, each batch can be regarded as a job given the batch forms. For convenience, the notations and variables are summarized as follows.

**Notations**

$n$: the number of jobs

$b$: the batch capacity

$t$: the processing time of batches on the batch machine

$w$: the limited waiting time

$p_i$: the processing time on a single machine for job $i$

**Variables**

$x_{ij}$: $x_{ij} = 1$ if job $i$ is assigned to batch $j$, otherwise, $x_{ij} = 0$, $i = 1,\ldots,n$, $j = 1,\ldots,n$

$y_{ij}$: $y_{ij} = 1$ if job $i$ is in batch $j$, and $p_i$ is the maximum processing time of batch $j$ on the single machine; otherwise, $y_{ij} = 0$

$s_j$: starting time of batch $j$ on the batch machine, $j = 1,\ldots,n$

$c_j$: completion time of batch $j$ on the single machine, $j = 1,\ldots,n$

$C_{\max}$: the makespan

On the basis of Lemma 1 and the presented variables, the problem can be formulated as the following mixed-integer linear programming model:

$$\min\ C_{\max} \tag{1}$$

$$\text{s.t. } \sum\nolimits_{j=1}^{n} x_{ij} = 1,\ \forall i = 1,...,n, \tag{2}$$

$$\sum\nolimits_{i=1}^{n} x_{ij} \leqslant b,\ \forall j = 1,...,n, \tag{3}$$

$$s_1 = 0, \tag{4}$$

$$s_j \geqslant s_{(j-1)} + tx_{(i,j-1)},\ \forall i = 1,...,n,\ j = 2,...,n, \tag{5}$$

$$c_j \geqslant s_j + tx_{ij} + \sum\nolimits_{k=1}^{n} x_{kj}p_k,\ \forall i = 1,...,n,\ j = 1,...,n, \tag{6}$$

$$c_j \geqslant c_{j-1} + \sum\nolimits_{i=1}^{n} x_{ij}p_i,\ \forall j = 2,...,n, \tag{7}$$

$$y_{ij} \leqslant x_{ij}, \forall i = 1,...,n,\ j = 1,...,n, \tag{8}$$

$$y_{ij} \geqslant x_{ij} - \sum\nolimits_{k=0}^{i-1} x_{kj},\ \forall i = 1,...,n,\ j = 1,...,n, \tag{9}$$

$$\sum\nolimits_{i=1}^{n} y_{ij} \leqslant 1,\ \forall j = 1,...,n, \tag{10}$$

$$c_j - \sum_{i=1}^{n} y_{ij} p_i - s_j - t \leqslant w, \ \forall j = 1,...,n, \quad (11)$$

$$C_{\max} \geqslant c_j \ \forall j = 1,...,n, \quad (12)$$

$$x_{ij} \in \{0,1\}, \ \forall i = 1,...,n, \ j = 1,...,n, \quad (13)$$

$$y_{ij} \in \{0,1\}, \ \forall i = 1,...,n, \ j = 1,...,n. \quad (14)$$

The objective function (1) is used to minimize the makespan. Constraint (2) ensures that one job can be assigned to only one batch. Constraint (3) guarantees that the number of jobs assigned to one batch cannot exceed the batch capacity. Constraint (4) indicates that the first batch can be started at time 0. Constraint (5) ensures that the start time of one batch should be no smaller than the standard completion time of the previous batch. Constraint (6) ensures that the completion time of one batch should be larger than or equal to the sum of the start time of the current batch, processing time on the batch machine, and total processing time on the single machine. Constraint (7) indicates that the completion time of one batch should be larger than or equal to the sum of the completion time of the previous batch and the total processing time on the single machine. Constraints (8)–(10) define the maximum processing time of jobs in one batch on the single machine. Constraint (8) ensures that $y_{ij} = 0$ if job $i$ is not in batch $j$. Constraint (9) ensures that $y_{ij} = 1$ when $x_{ij} - \sum_{k=0}^{i-1} x_{kj} = 1$, where $x_{0j} = 0$. Constraint (10) guarantees that one of $y_{ij}$ for each batch $j$ is equal to 1 for $i = 1,\ldots, n$ at most. Constraint (11) ensures that the waiting time of the jobs in one batch must be less than or equal to the limited waiting time $w$. Constraint (12) defines the makespan. Constraints (13) and (14) are the range of the variables.

The problem is NP-hard because the special case with $b = 1$ has been proven to be NP-hard (Yang and Chern, 1995). In this study, we focus on developing an efficient heuristic algorithm. The lower bounds and worst-case analysis are also provided.

## 3 Heuristic algorithm and lower bounds

The objective is to minimize the makespan. Each idle time of the single machine increases the makespan. The single machine demonstrates an idle time when the jobs of one batch are completed on the single machine before the subsequent batch is finished on the batch machine. Thus, we should balance the processing time on the batch and single machines by considering the batch capacity and limited waiting time constraints to avoid idle time. The proposed algorithm, denoted as balanced batch and single processing (BBSP), is described as follows:

BBSP

Step 1. Sort jobs in the nonincreasing order of $p_i$ to obtain a job list for $i = 1, 2, \ldots, n$.

Step 2. Assign the first job in the current list to the last position of batch $j = 1$. Remove this job from the list.

Step 3. Select the last job in the current list as the candidate for batch $j$.

- If the following conditions are met simultaneously, then assign this candidate job to batch $j$, remove this job from the list, and repeat Step 3.
  - The total number of jobs in batch $j + 1$ (the candidate job) is less than or equal to $b$.
  - The sum of the processing time on the single machine of jobs, excluding the job in the last position in batch $j$ and the processing time of the candidate job, is less than or equal to $w$.
  - The sum of the processing time on the single machine of jobs in batch $j$ is less than $t$.
- Else, $j = j + 1$ and repeat Step 2.

Step 4. If all of the jobs are assigned to batches, then proceed to Step 5.

Step 5. Calculate the sum of the processing time of jobs on the single machine as $P_j$ for each batch $j$. Then, the batches are processed in the nonincreasing order of $P_j$, $j = 1, 2, \ldots, n$.

Step 6. The jobs in one batch are processed consecutively on the single machine and start at the larger value between the completion time of the previous batch on the single machine and the completion time of this batch on the batch machine considering the limited waiting time constraint.

Step 7. Record the schedule $\pi_1$ and calculate the makespan $Obj(\pi_1)$.

Step 8. Adjust Step 3 as "Select the first job in the current list as the candidate for batch $j$." Repeat Steps 2–6 to obtain another schedule $\pi_2$. Calculate the makespan $Obj(\pi_2)$..

Step 9. Identify the index: $opt = \text{agrmin}_u Obj(\pi_u)$, $u = 1,2$. Return the schedule $\pi_{\text{opt}}$.

Three lower bounds are derived to test the performance of the BBSP algorithm. On one hand, the makespan includes the processing time of at least the first batch on the batch machine and the sum of the processing time of all jobs on the single machine. Thus, we can obtain the lower bound $LB_1 = t + \sum_{i=1}^{n} p_i$, directly. On the other hand, the makespan is no larger than the total batch processing time and the sum of the processing time of jobs that are in the last batch. The number of batches is no less than $\lceil n/b \rceil$. Thus, we can derive another lower bound, $LB_2 = \lceil n/b \rceil t + p_n$.

If the limited waiting time constraint is relaxed, then the relaxed problem can be solved optimally by the full batch–LPT policy (Ahmadi et al., 1992). This rule works as follows: the jobs are sorted in the nonincreasing order of $p_i$, $i = 1,2,\ldots,n$ and formed full batches continuously as much as possible. Then, the batch with the first $b$ LPT on the single machine is processed first on the batch machine. The

makespan of the relaxed problem achieved by the full batch–LPT policy is a lower bound of the investigated problem, which is denoted as $LB_3$.

These three lower bounds are used to test the efficiency of the proposed algorithm. In the numerical experiments, the maximum value of the three lower bounds, $LB = \max\{LB_1, LB_2, LB_3\}$, is used as the final test measurement.

Then, we present an upper bound of the proposed algorithm to conduct the worst-case analysis in Lemma 2.

**Lemma 2.** We let $Z$ represent the objective value achieved by the BBSP algorithm. Then, we derive:

$$Z \leqslant nt + \sum_{i=1}^{n} p_i.$$

**Proof.** If we let each batch consist of only one job and schedule it using the LPT first rule, then we can obtain another feasible schedule $\pi_3$ and denote the related makespan as $Obj(\pi_3)$. The job sequence on the single machine of schedule $\pi_3$ is the same as the schedule $\pi_2$ of the proposed algorithm. However, given that the completion time on the batch machine for each job in $\pi_3$ becomes less than or equal to that of $\pi_2$, we derive $Z \leqslant Obj(\pi_2) \leqslant Obj(\pi_3)$.

For $Obj(\pi_3)$, we assume that $p_1 \geqslant \ldots \geqslant p_k \geqslant t \geqslant p(k+1) \geqslant \ldots \geqslant p_n$, where $0 \leqslant k \leqslant n$. For $1 < i \leqslant n$, $k \geqslant 2$, we increase the processing time of each batch to the same as the processing time on the single machine and we derive:

$$Obj(\pi_3) \leqslant t + \sum_{i=1}^{k-1} p_i + \sum_{i=k+1}^{n} t + p_n \leqslant nt + \sum_{i=1}^{n} p_i.$$

When $k = 0$ or $k = 1$, $Obj(\pi_3) \leqslant nt + \sum_{i=1}^{n} p_i$. This equation completes the proof.

On the basis of the lower and upper bounds, we formulate Theorem 1.

**Theorem 1.** The worst-case ratio of the BBSP algorithm is bounded by $b + 1$, where $b$ is the batch capacity.

**Proof.** We let $Z^*_{\text{opt}}$ denote the optimal objective value and $Z$ represent the objective value achieved by the BBSP algorithm. Then, we derive

$$\frac{Z}{Z^*_{\text{opt}}} \leqslant 1 + \frac{UB - LB_1}{LB_2} \leqslant 1 + \frac{(n-1)t}{\left\lceil \frac{n}{b} \right\rceil t} \leqslant 1 + \frac{nt}{n/b^t} = 1 + b.$$

This equation completes the proof.

The BBSP algorithm can obtain the optimal solution for the problem when the processing time of the batches on the batch machine is less than or equal to the smallest processing time of jobs on the single machine.

**Lemma 3.** When $p_n > t$, the BBSP algorithm provides an optimal schedule for the problem.

**Proof.** When $p_n > t$, the upper bound achieved by assigning each batch with one job can be calculated as $t + \sum_{i=1}^{n} p_i$, which is equal to the lower bound $LB_1$, indicating that the BBSP algorithm provides an optimal schedule for the problem.

## 4 A HDE-based method

The HDE algorithm integrating the solution of the BBSP algorithm is proposed to further improve the solution quantity. Differential evolution (DE) was proposed by Storn and Price (1997) and is widely applied to various optimization problems (Fu et al., 2012; Wang et al., 2013; Shao and Pi, 2015; Zhang et al., 2017). Several notations and operators used in DE are briefly presented in this section. Fu et al. (2012) may be reviewed for the details.

**Notations**

- *TimeLimit*: the maximum running time;
- *MaxSame*: the maximum number of successive generations with the same objective value;
- *PS*: the population size;
- best($g$): the best objective value at generation $g$;
- $X_i(g)$:the $i$th individual of the $D$-dimensional search space at generation $g$;
- $X_i(g) = [\sigma_{i,1}(g), \sigma_{i,2}(g), \ldots, \sigma_{i,D}(g)]$;
- $V_i(g+1)$: a mutant individual for each target individual $X_i(g)$;
- $U_i(g+1)$: a trial individual for each target individual $X_i(g)$ and the corresponding mutant individual $V_i(g+1)$;
- $\lambda \in [0,1], F \in [0,2], CR \in [0,1]$: three control variables;
- rand($j$): the $j$th random number, which is uniformly distributed from [0,1];
- rand$n(i)$: a randomly selected index from the set of $\{1,2,\ldots,D\}$;
- $f(\blacksquare)$: the objective value.

**Operators**

Mutation operator

$$V_i(g+1) = X_\alpha(g) + \lambda^*(\text{best}(g) - X_\alpha(g) + F^*(X_\beta(g) - X_\gamma(g)). \tag{15}$$

Crossover operator

$$u_{i,j}(g+1) = \begin{cases} u_{i,j}(g+1) & \text{if } (\text{rand}(j) \leqslant CR) \text{ or } j = \text{rand}n(i) \\ x_{i,j}(g+1) & \text{otherwise} \end{cases}. \tag{16}$$

Selection operator

$$X_i(g+1) = \begin{cases} U_i(g+1) & \text{if } f(U_i(g+1)) < f(X_i(g)) \\ x_i(g) & \text{otherwise} \end{cases}. \tag{17}$$

The code value in each job is a real number from [0, 1]. Then, the jobs are sequenced in nonincreasing order of the code values. The batches are formed greedily given a job sequence. From the first job, jobs are continuously assigned to each batch as much as possible considering the batch capacity and limited waiting time constraints.

Then, the batches are processed in the nonincreasing order of the sum of the processing time of jobs in each batch. The procedures of the HDE algorithm are described as follows:

HDE
Step 1. Initialization
• Set $g = 0$;
• Generate $PS$–1 individuals randomly and integrate the solution of the BBSP algorithm as one individual;
• Calculate the objective value of each individual;
• Calculate best($g$).
Step 2. Evolution
• For each individual,
– Use mutation operator (15) to obtain a mutant individual;
– Use crossover operator (16) to obtain a trial individual;
– Calculate the objective value of this individual;
– Use selection operator (17) to obtain the individual of the next generation.
• End for
• Set $g = g + 1$ and update best($g$) and the best solution.
Step 3. Termination
• If the *TimeLimit* or *MaxSame* is reached, then proceed to Step 4;
• Else, repeat Step 2.
Step 4. Return the solution with the best objective value.

The worst-case analysis in Theorem 1 is also applicable to the HDE algorithm because the HDE algorithm integrates the solutions of the BBSP algorithm.

## 5 Numerical experiments

In this section, the computational experiments are conducted to test the performances of the proposed formulation and algorithm. All the tests are run on a 64bit Windows 10 platform with Intel Core 3.2 GHz CPU and 8.0 GB RAM. The mathematical formulation is solved using CPLEX 12.6 under default configuration, with a time limit of 3600 s. The heuristic algorithms are coded in MATLAB.

### 5.1 Problem instance generation

The problem case is presented in the form of combination "$n$–$b$," where $n$ is the number of jobs and $b$ is the batch capacity. As used by Su (2003) and Zhang et al. (2017), we present the instance generation rule as follows:

The processing time on the single machine is an integer randomly generated from the uniform distribution [1, 10]. We let the processing time of one batch in the batch machine be an integer randomly generated from the uniform distribution $\left[10, \frac{b}{n}\sum_{i=1}^{n} p_i + 10\right]$. because the batch machine is time-consuming. We denote the sum of the first $b$ LPT on the single machine as $P_{\max}$. If the limited waiting time is larger than, $P_{\max}$, then the limited waiting time constraint is pointless. Thus, we let the limited waiting time be an integer randomly generated from the uniform distribution $\left[\frac{b}{n}\sum_{i=1}^{n} p_i, P_{\max}\right]$. We let $G_1$ and $G_2$ denote two sizes of instances. The configuration of each group is expressed as follows:

$$G_1 : n = \{8,\ 12,\ 16,\ 20\} \text{ and } b = \{2,\ 4\},$$

$$G_2 : n = \{50,100,200,400\} \text{ and } b = \{10,\ 20\}.$$

For each combination, we generate 20 instances to test the performance of the proposed algorithm.

The values of the parameters used in the HDE algorithm are presented as follows:

- $TimeLimit = 600$ s;
- $MaxSame = 30$;
- $PS = 2n,\ D = n,\ CR = 0.5,\ F = 0.7,\ \lambda = 0.4$.

### 5.2 Numerical results and analysis

Table 1 lists the problem size that can be solved to optimality with the first 10 instances in each combination. The #*Opt* column shows the number of instances solved to optimality within 1 h by the mixed integer linear programming model. The *Time* column shows the average time for the instances solved to optimality by the mathematical model. The *Gap* column shows the gap between the results of the HDE algorithm and optimal solutions. When no instance in one combination is solved to optimality within 1 h by using the formulation, #*Opt* is zero and the *Time* and *Gap* columns display "–." In the table, the problem size that can be solved to optimality by the mathematical model is reported as 12 jobs within 1 h on a single PC. The *Gap* column shows that the HDE algorithm can also obtain optimal solutions.

Table 2 presents the results for instances in $G_1$. The $n$–$b$ column represents the combination between the number of

**Table 1** Comparison with optimal solutions of instances in $G_1$

| *n*–*b* | #*Opt* | *Time*/s | *Gap*/% |
|---|---|---|---|
| 8–2 | 10 | 2.07 | 0.00 |
| 8–4 | 10 | 0.16 | 0.00 |
| 12–2 | 2 | 1.65 | 0.00 |
| 12–4 | 10 | 9.08 | 0.00 |
| 16–2 | 0 | – | – |
| 16–4 | 4 | 36.54 | 0.00 |
| 20–2 | 0 | – | – |
| 20–4 | 0 | – | – |

jobs and the batch capacity. The *AvgG-HDE* column displays the average gap between the objective value achieved by the HDE algorithm and the lower bound. The *MaxG-HDE* column shows the maximum gap between the objective value achieved by the HDE algorithm and the lower bound. The *T-HDE* column presents the average time (in seconds) for the HDE algorithm. The *AvgG-Su* column indicates the average gap between the objective value achieved by the algorithm proposed by Su (2003) and the lower bound. The *T-Su* column shows the average time (in seconds) for the algorithm proposed by Su (2003). We first calculate the gap and record the time for each instance in this combination when calculating the average gap and time for one combination. Then, we calculate the average gap and time.

Table 2 indicates that the solution times used by the HDE algorithm and the algorithm proposed by Su (2003) are all minimal. The maximum gap for the HDE algorithm is 0.65% in the combination 20–4. For the other combinations, the HDE algorithm can obtain optimal solutions. These results show that the HDE algorithm has a good performance for the instances in small-sized problems. All of the average gaps are smaller in the HDE algorithm than in the algorithm proposed by Su (2003). This finding indicates that the HDE algorithm can obtain better solutions than the algorithm proposed by Su (2003).

The $t$ test is conducted to verify this observation. We establish the null hypothesis $H_0$ as $AvgG–HDE–AvgG–Su \geqslant 0$ and the alternative hypothesis $H_1$ as $AvgG–HDE–AvgG–Su < 0$. Table 2 displays that all of the p-values are less than the 5% significance level, which supports the observation that the HDE algorithm is more competitive than the algorithm proposed by Su (2003) for instances in $G_1$.

Table 3 shows the results for instances in $G_2$. The columns are similar to the columns in Table 2. The maximum gap for the HDE algorithm is 3.49% in the combination 200–20. For combinations 50–20 and 100–20, all of the instances are solved to optimality. The results of the $t$ test show that the HDE algorithm can obtain better solutions than the algorithm proposed by Su (2003) for instances in $G_2$.

## 6 Conclusions

In this study, we analyze the two-stage flow shop scheduling problem including batch and single machines. Limited waiting constraint is considered to minimize the makespan. A mixed-integer linear programming model is proposed for this problem. Three tight lower bounds and a heuristic algorithm are developed. The worst-case performance of the proposed algorithm is discussed. The HDE algorithm integrating the solution of the heuristic algorithm is developed to improve the solution quantity. The

**Table 2** Performance of the HDE algorithm for instances in $G_1$

| *n–b* | *AvgG-HDE*/% | *MaxG-HDE*/% | *T-HDE*/s | *AvgG-Su*/% | *T-Su*/s | *p*-value |
|---|---|---|---|---|---|---|
| 8–2 | 0.00 | 0.00 | 0.03 | 3.26 | 0.00 | 6.76E-06 |
| 8–4 | 0.00 | 0.00 | 0.03 | 3.71 | 0.00 | 2.09E-04 |
| 12–2 | 0.00 | 0.00 | 0.05 | 2.42 | 0.00 | 1.13E-05 |
| 12–4 | 0.00 | 0.00 | 0.05 | 5.04 | 0.00 | 1.93E-06 |
| 16–2 | 0.00 | 0.00 | 0.08 | 1.60 | 0.00 | 6.68E-07 |
| 16–4 | 0.00 | 0.00 | 0.07 | 5.96 | 0.00 | 1.07E-07 |
| 20–2 | 0.00 | 0.00 | 0.10 | 1.04 | 0.01 | 8.08E-06 |
| 20–4 | 0.03 | 0.65 | 0.12 | 4.82 | 0.01 | 8.74E-09 |

**Table 3** Performance of the HDE algorithm for instances in $G_2$

| *n–b* | *AvgG-HDE*/% | *MaxG-HDE*/% | *T-HDE*/s | *AvgG-Su*/% | *T-Su*/s | *p*-value |
|---|---|---|---|---|---|---|
| 50–10 | 0.62 | 2.58 | 0.34 | 5.25 | 0.03 | 7.72E-08 |
| 50–20 | 0.00 | 0.00 | 0.26 | 1.97 | 0.02 | 5.09E-05 |
| 100–10 | 0.95 | 2.34 | 1.10 | 3.39 | 0.25 | 1.61E-08 |
| 100–20 | 0.00 | 0.00 | 0.79 | 1.91 | 0.16 | 1.01E-04 |
| 200–10 | 0.90 | 2.28 | 6.44 | 2.34 | 1.92 | 1.87E-07 |
| 200–20 | 1.00 | 3.49 | 4.36 | 3.17 | 1.23 | 3.94E-06 |
| 400–10 | 0.81 | 2.94 | 36.99 | 1.28 | 15.26 | 3.48E-02 |
| 400–20 | 1.12 | 2.09 | 19.52 | 2.16 | 9.68 | 2.23E-10 |

numerical results show that the proposed hybrid algorithm can obtain high-quality solutions and exhibit a competitive performance. Future work should be focused on developing a constant approximation algorithm for this problem to achieve a good computational performance.

**Acknowledgements** This research was supported in part by National Natural Science Foundation of China (Grant Nos. 71690232 and 71371015) and by National Science Foundation (Grant Nos. CMMI-1435800 and CMMI-1536978).

## References

Ahmadi J H, Ahmadi R H, Dasu S, Tang C S (1992). Batching and scheduling jobs on batch and discrete processors. Operations Research, 39(4): 750–763

Chung T P, Sun H, Liao C J (2016). Two new approaches for a two-stage hybrid flow shop problem with a single batch processing machine under waiting time constraint. Computers & Industrial Engineering, in press (https://doi.org/10.1016/j.cie.2016.11.031)

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

Gong H, Tang L, Duin C W (2010). A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times. Computers & Operations Research, 37(5): 960–969

Hoogeveen H, Velde S (1998). Scheduling by positional completion times: analysis of a two-stage flow shop problem with a batching machine. Mathematical Programming, 82(1–2): 273–289

Huang Z, Shi Z, Zhang C, Shi L (2017). A note on "Two new approaches for a two-stage hybrid flow shop problem with a single batch processing machine under waiting time constrain". Computers & Industrial Engineering, 110: 590–593

Shao W, Pi D (2015). A self-guided differential evolution with neighborhood search for permutation flow shop scheduling. Expert Systems with Applications, 51: 161–176

Storn R, Price K (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 flow shop with limited waiting time constraints. Computers & Industrial Engineering, 44(3): 409–424

Wang H Y, Lu Y B, Peng W L (2013). Permutation flow-shop scheduling using a hybrid differential evolution algorithm. International Journal of Computing Science and Mathematics, 4(3): 298–307

Yang D L, Chern M S (1995). A two-machine flow shop scheduling problem with limited waiting time constraints. Computers & Industrial Engineering, 28(1): 63–70

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