RAIRO-Oper. Res. 55 (2021) S253–S269
https://doi.org/10.1051/ro/2019108

RAIRO Operations Research
www.rairo-ro.org

OPTIMIZING BATCH-PROCESSING OPERATIONS WITH BATCH-POSITION-BASED LEARNING EFFECTS

Leilei Tai*

Abstract. Motivated by applications in porcelain-making companies, we consider a type of optimization problems for batch-processing operations. In production, a single batch-processing machine with a fixed capacity is used to process jobs. Several jobs can be processed together if their total size is no more than the machine capacity. Batch-position-based learning effects are considered because workers become skillful gradually after they perform the processing task repeatedly. The actual processing time of a batch is a decreasing function of its position in production. The objective is to minimize makespan and we consider three different problems. In the first problem, jobs have identical sizes and we present an algorithm which can find optimal solutions in polynomial time. In the second problem, jobs have identical processing times and we show the problem is NP-hard in the strong sense. We propose an approximation algorithm with an absolute performance guarantee of 1.5 and asymptotic performance guarantee of 1.223. In the third problem, jobs have non-identical sizes and processing times simultaneously. We propose an algorithm with an absolute and asymptotic performance guarantee of 2. Besides, we present the evolution of the performance guarantee and provide managerial insights for decision makers of manufacturing companies.

Mathematics Subject Classification. 90C97.

Received February 26, 2019. Accepted November 10, 2019.

1. Introduction

Learning effects have aroused a lot of interests of researchers in the recent two decades. By contrast to classical problems, the behavior of workers is considered in optimization problems with learning effects. That is, when a worker performs the same task repeatedly, he or she becomes skillful gradually and can finish a task more and more quickly. This is particularly true in the manufacturing companies with batch-processing machines and non-identical job sizes such as electroplating companies, porcelain companies, metal working companies, food making companies and so on. In this type of manufacturers, jobs have non-identical processing times and sizes. Batch-processing machines are used to process jobs and each machine has a fixed capacity. Jobs can be processed together in a batch as long as the total size of jobs does not exceed the machine capacity. The processing of a batch cannot be interrupted until all the jobs in it are completed, therefore, the processing time of a batch equals to the longest processing time of jobs in it. Since jobs need to be assigned in batches before being processed, the workers' ability is very important to improve the production efficiency. When a worker conducts the operations repeatedly, he or she can reduce the processing time gradually.

Keywords. Optimization, learning effects, batch processing, scheduling, heuristics.

Anhui University of Traditional Chinese Medicine, Hefei 230012, P.R. China.
*Corresponding author: 8499640@qq.com

Take the porcelain companies for example. The key operation of the production of porcelains is the calcination, which is conducted in a calcination oven. Learning effects are important to improve the efficiency of calcination. First, the workers need to check the semi-products and judge whether they are qualified to be processed. The semi-products are made of clay and have different sizes. A Skillful worker can finish the judgement in a short time but an apprentice often spends a longer time. Then the semi-products are assigned in batches, which are processed one by one in the oven. The oven has a large size to accommodate semi-products, which are burned in at a high temperature around 2000 degrees Fahrenheit and the calcination cannot be interrupted until all the semi-products are processed. When filling a batch of semi-products in the oven, each semi-product should be put in a proper position to improve the product quality and energy utilization. A skillful worker can also finish the filling operation fast but a new worker often needs more time. Finally, skillful workers can make a correct decision on the completion time, that is, the time when all the semi-products are completed but a new worker cannot. By the above introduction, we find that the workers' ability is important to improve makespan and a worker is trained by the operations of batches. Skills of assigning a batch, filling a batch and processing a batch are key operations in the calcination. Usually a worker can performs better gradually after processing batches repeatedly. Therefore, we propose *batch-position-based learning effects* for this type of manufacturers. The objective is to minimize makespan with batch-position-based learning effects.

The current research focuses on the classical scheduling [21], *i.e.*, a machine processes one job or a fixed number of jobs at a time. Biskup [2] proposes the scheduling model with learning effects where $t_j^A = t_j l^\alpha$. t_j represents the normal processing time of job j, t_j^A represents the actual processing time if j is assigned as the lth job to be processed in production. α represents the learning index and $\alpha < 0$. DeJong's learning effects [5] are also widely used in the research of scheduling and the processing time is defined as $t_j^A = t_j(M + (1 - M)l^\alpha)$, where $0 \leq M \leq 1$ and l is the position of job j in production, *i.e.*, j is the lth job to be processed. Koulamas [12] studies the single-machine scheduling problem with learning effects and provides an optimal algorithm to minimize makespan. Lee *et al.* [15] consider the release times in the single-machine problem and provide a branch-and-bound algorithm. Qian and Steiner [22] consider both the learning effects and deterioration effects and propose polynomial-time optimal algorithms. Jiang *et al.* [11] consider the learning effects determined by actual processing times and job positions, where Problems of minimizing $C_{\max}$ and $\sum_{j=1}^n C_j$ are shown to be NP-hard in the ordinary sense and problem of minimizing maximal lateness is shown to be NP-hard in the strong sense. Except for the research on single-machine problems, multi-machine problems with learning effects are also studied, such as parallel-machine problems [9, 20] and flowshop problems [14].

However, in the problem of scheduling batch-processing machines with jobs having non-identical sizes, jobs are processed in batches and the number of jobs in a batch is not a constant. As introduced in the beginning of this section, several jobs can be processed together as long as their total size is not more than the machine capacity. Obviously, this type of scheduling is more difficult to optimize. The problem of minimizing makespan on a single batch-processing machine with non-identical sizes is NP-hard in the strong sense [3]. Akiyoshi *et al.* [1] study the problem with controllable processing times and propose a recursive decomposition algorithm to minimize the production cost. Giglio [7] studies the problem with family constraint and proposes mixed-integer linear programming models. Wang and Leung [27] study the parallel-machine problems with non-identical capacities and propose algorithms to minimize makespan.

Cheng *et al.* [4] study the joint scheduling of production and distribution to minimize service span, *i.e.*, the period lasting from the beginning of production to the end of product distribution. In practice, scheduling batch-processing machines is widely applied in steel making companies. Wiebke *et al.* [28] consider the problem of minimizing makespan for a given set of coils and a graph theoretic model is proposed. Tang *et al.* [25] consider the problem of integrated charging and casting problem in steel making and propose optimization method on minimizing production cost. Tan *et al.* [23, 24] consider the integrated model of digital goods. Li and Petruzzi [17] consider the model with demand uncertainty. Then, Liu *et al.* [19], Hidayat *et al.* [8] and Li [16] consider the parallel-machine model to minimize the maximal lateness, which is NP-hard in the strong sense. However, no research has been found on the scheduling of batch-position-based learning effects.

In this paper, we consider the batch-position-based learning effects on a single batch-processing machine and propose three problems. Polynomial time algorithms are proposed and the performance guarantees are analyzed. The remainder of this paper is organized as follows. In Section 2, we present the scheduling problem with batch-position-based learning effects and show the computational complexity. In Section 3, we propose an algorithm for the problem with identical job sizes to find optimal solutions. In Section 4, we propose an approximation algorithm for the problem with identical processing times. In Section 5, we consider the general problem and propose an approximation algorithm. In Section 6, we conclude this paper and present directions for future research.

2. Problem description

The problem under investigation can be described as follows. There are n jobs to be processed and the job set is $J = \{1, \ldots, n\}$. Job j has a processing time of t_j and a size of s_j. In production, jobs are assigned in batches and the batch set is $B = \{b_1, \ldots, b_K\}$, where K is the number of batches and b_k represents the kth batch. The machine capacity is D and for any batch b_k, $\sum_{j\in b_k} s_j \le D$. The normal processing time of b_k is

$$T_k = \max\{t_j : j \in b_k\}, \tag{2.1}$$

i.e., the processing of a batch cannot be interrupted until all the jobs are completed. Because of the learning effects, the workers become more and more skillful and the actual processing times of the latter batches are shorter than their normal processing times. The actual processing time of b_k is defined as

$$T_k^A = (M + (1 - M)k^\alpha)T_k. \tag{2.2}$$

Here $0 \le M \le 1$ and is a constant. α is the learning index and $\alpha < 0$. k is the position of batch b_k and the learning effects are called batch-position-based learning effects. The makespan is

$$\begin{aligned} C_{\max} &= \sum_{k=1}^{K} T_k^A = \sum_{k=1}^{K} (M + (1 - M)k^\alpha)T_k \\ &= M\sum_{k=1}^{K} T_k + (1 - M)\sum_{k=1}^{K} k^\alpha T_k \\ &= M\gamma + (1 - M)\delta, \end{aligned} \tag{2.3}$$

where $\gamma = \sum_{k=1}^{K} T_k$ and $\delta = \sum_{k=1}^{K} k^\alpha T_k$ for simplicity. Since $\alpha < 0$, we have $\gamma > \delta$. We denote the learning effects as BLE. Let $t_{\min} = \min\{t_j : j \in J\}$ and $t_{\max} = \max\{t_j : j \in J\}$. Define r as

$$r = \frac{t_{\min}}{t_{\max}}, \tag{2.4}$$

where $0 < r \le 1$ and a larger r means a wider difference between the maximal and minimal processing time. We consider three different problems in this paper where the production constraints are different. A scheduling problem can be denoted as $\eta_1|\eta_2|\eta_3$, where η_1 represents the machine configuration, η_2 represents the production constraints and η_3 represents the objective function. The problems under investigation can be denoted as follows.

$\Pi_1 : 1|\text{BLE}, \text{batch}, s_j = 1, t_j|C_{\max}$
$\Pi_2 : 1|\text{BLE}, \text{batch}, s_j, t_j = 1|C_{\max}$
$\Pi_3 : 1|\text{BLE}, \text{batch}, s_j, t_j|C_{\max}$

In the three problems, there is one batch-processing machine and jobs can be processed in batches. Batch-position-based learning effects are considered and the objective is to minimize makespan. But the production constraints are different. In Π_1, jobs have the same size of 1 but non-identical processing times while in Π_2, jobs have non-identical sizes and unit processing times. Π_3 is the general problem where jobs have non-identical sizes and non-identical processing times at the same time. Next we analyze the computational complexity of the problems.

We will present an optimal polynomial-time algorithm for Π_1 in Section 3. Now we analyze the computational complexity of Π_2. Consider a special case Π_2^s where $M = 1$. Cheng *et al.* [3] have shown that Π_2^s is NP-hard in the strong sense. Since Π_2^s is a special case of Π_2 which is also a special case of Π_3, in computational complexity $\Pi_2^s \propto \Pi_2 \propto \Pi_3$. We have the following proposition.

Porposition 2.1. *Π_2 and Π_3 are both NP-hard in the strong sense.*

3. Algorithm for Π_1

In the following content, we use Ai and $\pi_i (i = 1, 2, 3)$ to denote our algorithm for solving Π_i and the corresponding solution, respectively. For simplicity, we use "*" as the optimality operator. For example, π_1^* represents an optimal solution of Π_1 and K^* represents the number of batches in an optimal solution. In this section, we consider problem Π_1, *i.e.*, $1|\text{BLE, batch}, s_j = 1, t_j|C_{\max}$. We propose the following Algorithm 3.1 for Π_1.

Algorithm 3.1.

Step 1. Order the jobs in non-increasing order of their processing times t_j and relabel them as $1, \ldots, n$.
Step 2. Assign the first D jobs in the first batch B_1. Then repeat to assign the next D jobs in a batch until the number of remaining jobs is no more than D. Then assign them in the last batch. The obtained batch set is $\{B_1, \ldots, B_K\}$.
Step 3. Order the batches in non-decreasing order of their processing times and relabel them as $b_1, \ldots, b_K$. Process them one by one in the order.

Algorithm 3.1 finds an optimal number of batches because each job has a size of 1, the optimal number of batches is $K^* = \lceil n/D \rceil$. Observe Steps 1 and 2 of 3.1 and we have $K = \lceil n/D \rceil$, *i.e.*, $K = K^*$.

Lemma 3.2. *If we order the batches of π_1^* in non-decreasing order of their processing times and relabel them as $b_1^*, \ldots, b_{K^*}^*$, then $T_k \leq T_k^*$ for $k = 1, \ldots, K^*$.*

Proof. In Figure 1, π_1 are shown on the left side, where a batch is represented by a large rectangle and a job is represented by a small rectangle in it. The length and the height of a rectangle denote the processing time and the size respectively. In Figure 1, small rectangles have the same height since all jobs have the same size. If any change is made on π_1, without loss of generality, suppose jobs x and y are swapped, where $y \in b_g$ and $x \in b_h$ satisfying $t_y < t_x$ and $g < h$. The obtained solution is π_1^s as shown on the right side of Figure 1. By (2.1), $T_g \geq t_y$ and $T_h \geq t_x$. Since batches are ordered in non-decreasing order of processing times and $g < h$, we have $T_g \leq T_h$.

Now we consider the following four cases.

Case 1. $t_y = T_g$ and $t_x = T_h$, that is, y is the longest job in batch b_g and x is the longest job in b_h. In this case, after swapping x and y we have $T_g = t_x = T_h^s$ and $T_h = t_x = T_y^s$. So $T_g + T_h \leq T_g^s + T_h^s$.
Case 2. $t_y = T_g$ and $t_x < T_h$, that is, y is the longest job in b_g but x is not the longest job in b_h. After swapping we have $T_g = t_y < t_x = T_g^s$ and $T_h = T_h^s$.
Case 3. $t_y < T_g$ and $t_x = T_h$. After swapping we have $T_g \leq T_h^s$ and $T_h = T_x = T_g^s$.
Case 4. $t_y < T_g$ and $t_x < T_h$. After swapping we have $T_g < t_x = T_g^s$ and $T_h = T_h^s$.

By the above analysis, we find in all the cases,

$$T_g + T_h \leq T_g^s + T_h^s. \tag{3.1}$$

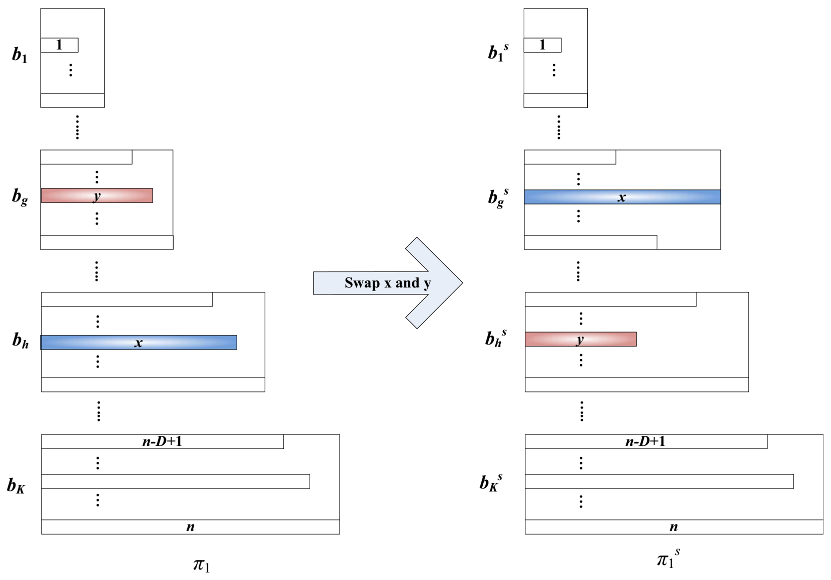

FIGURE 1. Swapping any two jobs in π_1.

Since $C_{\max} = \sum_{k=1}^{K} T_k$, we have $C_{\max} \le C^s_{\max}$. It is easy to see that if x and y are in the same batch, $C_{\max} = C^s_{\max}$. So any change on π_1 cannot improve the makespan.

Obviously, If more than two jobs are swapped, Lemma 3.2 also holds in the same way. This completes the proof. □

Lemma 3.3. *Given a batch set $\{b_1, \ldots, b_K\}$, processing them in non-decreasing order of processing times minimizes makespan.*

Proof. Since the batches are given, they can be handled as jobs. By Lemma 3.2,

$$\gamma = \sum_{k=1}^{K} T_k = \sum_{k=1}^{K^*} T_k \le \sum_{k=1}^{K^*} T_k^* = \gamma^*. \tag{3.2}$$

Now we show $\delta = \sum_{k=1}^{K} k^\alpha T_K^A$ is also optimal by contradiction. In π_1, batches are ordered in non-decreasing order of processing time, *i.e.*, $T_1 \le T_2 \le \cdots \le T_K$. Suppose b_g and b_h are interchanged and the new solution is π'. Without loss of generality, let $g < h$. Then

$$\begin{aligned}
\delta - \delta' &= \sum_{k=1}^{K} k^\alpha T_k^A - \left(\sum_{k=1}^{K} k^\alpha T_k^A\right)' \\
&= \left(\sum_{k=1}^{g-1} k^\alpha T_k + T_g g^\alpha + \sum_{k=g+1}^{h-1} T_k k^\alpha + T_h h^\alpha + \sum_{k=h+1}^{K} T_k k^\alpha\right) \\
&\quad - \left(\sum_{k=1}^{g-1} k^\alpha T_k + T_h g^\alpha + \sum_{k=g+1}^{h-1} T_k k^\alpha + T_g h^\alpha + \sum_{k=h+1}^{K} T_k k^\alpha\right) \\
&= T_g g^\alpha + T_h h^\alpha - T_h g^\alpha - T_g h^\alpha \\
&= (T_g - T_h)(g^\alpha - h^\alpha) \le 0
\end{aligned} \tag{3.3}$$

since $\alpha < 0$ and $0 < g < h$. That is,

$$\delta \leq \delta'. \tag{3.4}$$

By (2.3), (3.3) and (3.4)

$$C_{\max} = M\gamma + (1-M)\delta \leq M\gamma' + (1-M)\delta' = C'_{\max}, \tag{3.5}$$

which implies that no change on π_1 can improve makespan. Obviously if more than one change is made on π_1, the makespan cannot be improved. Lemma 3.3 follows. □

By Lemmas 3.2 and 3.3, we see 3.1 finds an optimal batch set and assigns the batches in an optimal way, therefore, 3.1 finds an optimal $C_{\max}$. Now we analyze the time complexity of 3.1. Step 1 is to order n jobs and the time complexity is $O(n \log n)$. Step 2 is to assign n jobs in batches and the time complexity is $O(n)$. Step 3 is to order K batches and the time complexity is $O(n \log n)$ since $K \leq n$. Therefore, the performance of 3.1 is as follows.

Theorem 3.4. *Algorithm 3.1 finds an optimal makespan for Π_1 in $O(n \log n)$ time.*

4. Approximation algorithm for Π_2

In this section, we consider Π_2, *i.e.*, $\Pi_2 : 1|\text{BLE}, \text{batch}, s_j, t_j = 1|C_{\max}$. Jobs have non-identical sizes but the same processing time of 1. As shown in Proposition 1, the problem is NP-hard in the strong sense. We propose the following Algorithm 4.1.

Algorithm 4.1.

Step 1. Order the jobs in non-increasing order of their sizes and relabel them as $1, \ldots, n$.
Step 2. Assign jobs into batches using First Fit rule. Create the first empty batch b_1 and put job 1 in it. Then check all the following jobs one by one as follows. If the total size of jobs is no more than D when putting j in b_1, then assign j in b_1. Otherwise go to check the next job. When all the jobs are checked, b_1 is obtained. Then create the second empty batch b_2 and check the remained jobs. Repeat creating empty batches until there is no job left. The obtained batch list is $b_1, \ldots, b_K$.
Step 3. Process batches in the order of $b_1, \ldots, b_K$.

Steps 1 and 2 are equivalent to the famous First Fit Decreasing rule for Bin Packing Problem. Dosa *et al.* [6] show that the bound of FFD is $N_{\text{FFD}} \leq \frac{11}{9}N_{\text{OPT}} + \frac{2}{3}$, where N_{FFD} represents the number of bins obtained by FFD and N_{OPT} represents the optimal number of bins. By the conversion before Proposition 2.1 in Section 2, we have the following proposition.

Porposition 4.2. *Algorithm 4.1 finds a solution π_2 with $K \leq \frac{11}{9}K^* + \frac{2}{3}$.*

In Π_2, each job has a unit processing time and so $T_k = \max\{t_j : j \in b_k\} = 1$. By (2.3),

$$\gamma = \sum_{k=1}^{K} T_k = K \tag{4.1}$$

and

$$\delta = \sum_{k=1}^{K} k^\alpha T_k = \sum_{k=1}^{K} k^\alpha. \tag{4.2}$$

Lemma 4.3. *In π_2, $\gamma \leq 1.5\gamma^*$, i.e., $\sum_{k=1}^{K} T_k \leq 1.5 \sum_{k=1}^{K^*} T_k^*$.*

Proof. We analyze γ and γ^* in the following three cases where $K^* = 1$, $K^* = 2$ and $K^* \geq 3$, respectively.

Case 1: $K^* = 1$.

This implies that all the jobs can be assigned in one batch, that is, $\sum_{j=1}^n s_j \leq D$. In this case, π_2 is optimal since jobs can also be assigned in one batch. Thus

$$\frac{\gamma}{\gamma^*} = 1. \tag{4.3}$$

Case 2: $K^* = 2$.

In this case, there are 2 batches in π_2^*. By Proposition 4.2, $K \leq \frac{22}{9} + \frac{2}{3} = \frac{28}{9}$, that is, $K = 2$ or 3. If $K = 2$, then $\gamma = \gamma^*$. If $K = 3$, then $\gamma = 1.5\gamma^*$. So when $K^* = 2$,

$$\frac{\gamma}{\gamma^*} \leq 1.5 \tag{4.4}$$

and the worst case appears when $K = 3$.

Case 3: $K^* \geq 3$.

In this case, we have

$$\frac{\gamma}{\gamma^*} = \frac{K}{K^*} \leq \frac{\frac{11}{9}K^* + \frac{2}{3}}{K^*} = \frac{11}{9} + \frac{2}{3K^*} \leq \frac{11}{9} + \frac{2}{9} = \frac{13}{9} = 1.445. \tag{4.5}$$

By (4.3)–(4.5), we have $\gamma/\gamma^* \leq 1.5$. □

Now we consider δ/δ^*, *i.e.*, $\sum_{k=1}^K k^\alpha / \sum_{k=1}^{K^*} k^\alpha$. We also investigate three cases as shown in the proof of Lemma 4.3. When $K^* = 1$, there is no learning effect since there is only one batch in π_2. That is,

$$\frac{\delta}{\delta^*} = \frac{1^\alpha}{1^\alpha} = 1. \tag{4.6}$$

When $K^* = 2$, $K = 2$ or 3. If $K = 2$, $\delta/\delta^* = 1$. If $K = 3$,

$$\frac{\delta}{\delta^*} = \frac{1^\alpha + 2^\alpha + 3^\alpha}{1^\alpha + 2^\alpha} = 1 + \frac{3^\alpha}{1 + 2^\alpha}.$$

Consider function $f(\alpha) = \frac{3^\alpha}{1+2^\alpha}$ and we have

$$f'(\alpha) = \frac{3^\alpha(1 + 2^\alpha)\log 3 - 3^\alpha 2^\alpha \log 2}{(1 + 2^\alpha)^2} > 0.$$

That is, $f(\alpha)$ is an increasing function of α. Since $\alpha < 0$, we have $f(\alpha) < f(0) = \frac{1}{2}$. So when $K^* = 2$,

$$\frac{\delta}{\delta^*} = 1 + f(\alpha) < 1 + 0.5 = 1.5. \tag{4.7}$$

When $K^* \geq 3$,

$$\begin{aligned}
\frac{\delta}{\delta^*} &= \frac{\sum_{k=1}^K k^\alpha}{\sum_{k=1}^{K^*} k^\alpha} \\
&= \frac{1^\alpha + \cdots + K^\alpha}{1^\alpha + \cdots + (K^*)^\alpha} = 1 + \frac{(K^*+1)^\alpha + \cdots + K^\alpha}{1^\alpha + \cdots + (K^*)^\alpha} \\
&\leq 1 + \frac{(K^*+1)^\alpha (K - K^*)}{(K^*)^\alpha K^*} \leq 1 + \frac{K - K^*}{K^*} = \frac{K}{K^*} \\
&\leq \frac{\frac{11}{9}K^* + \frac{2}{3}}{K^*} = \frac{11}{9} + \frac{2}{3K^*} \leq \frac{11}{9} + \frac{2}{9} = \frac{13}{9} \\
&= 1.445.
\end{aligned} \tag{4.8}$$

By the above discussion, we have the following lemma on δ.

Lemma 4.4. *In π_2, $\delta/\delta^* \leq 13/9 = 1.5$, i.e., $\sum_{k=1}^{K} k^\alpha T_k \leq 1.5 \sum_{k=1}^{K^*} k^\alpha T_k^*$.*

Now we obtain the performance of 4.1 when solving Π_2.

Theorem 4.5. *The time complexity of 4.1 is $O(n^2)$. When solving Π_2, Algorithm 4.1 finds a solution with $C_{\max}/C^*_{\max} \leq 1.5$ for all instances. When the size of Π_2 approaches infinity, $\lim_{n\to\infty} C_{\max}/C^*_{\max} = 11/9 = 1.223$*

Proof. The time complexity of 4.1 can be analyzed as follows. Steps 1 is to order jobs and batches and they both cost $O(n \log n)$ time. In Step 2, testing operations are needed for a job. Since there are n jobs, the number of testing operations is no more than n for each job. Step 3 is executed in $O(n)$ time. Therefore, the time complexity of 4.1 is $O(n^2)$.

By Lemmas 4.3 and 4.4, we have $C_{\max}/C^*_{\max} \leq 1.5$ for all instances. The worst case of 1.5 appears in instances where $K^* = 2$ and $K = 3$, however, in all the other cases, $C_{\max}/C^*_{\max} < 1.5$.
Response: By (4.5) and (4.8), we have $\gamma/\gamma^* \leq \frac{11}{9} + \frac{2}{3K^*}$ and $\delta/\delta^* \leq \frac{11}{9} + \frac{2}{3K^*}$. So when the size of Π_2 approaches infinity,

$$\lim_{n\to\infty} \frac{C_{\max}}{C^*_{\max}} = \lim_{K^*\to\infty} \frac{C_{\max}}{C^*_{\max}} \leq \lim_{K^*\to\infty} \frac{11}{9} + \frac{2}{3K^*} = \frac{11}{9}. \tag{4.9}$$

Theorem 4.5 follows. □

By (2.3), $C_{\max} = M\sum_{k=1}^{K} T_k + (1-M)\sum_{k=1}^{K} k^\alpha T_k$ where $0 \leq M \leq 1$. When M decreases, the leaning effects become more important in the production. If $M = 0$, we find $C_{\max} = \sum_{k=1}^{K} k^\alpha T_k$. Observe the proof of Lemma 4.4, we have $C_{\max}/C_{\max^*} \leq 1.445$ which is better than the ratio in Theorem 4.5. This indicates that learning effects have positive influence on the productivity. If workers are well trained and perform in a good way, they can improve the production system effectively. However, by (4.8), we find that if α decreases, Algorithm 4.1 can find a better objective function. A smaller α means workers are better in learning and working, so better workers can make a better production performance, *i.e.*, a smaller makespan.

5. Approximation algorithm for the general problem

Now we consider the general problem Π_3, *i.e.*, $1|\text{BLE}, \text{batch}, s_j, t_j|C_{\max}$. In this problem, jobs have non-identical sizes and processing times at the same time and the problem is NP-hard in the strong sense as shown in Proposition 2.1. We propose the following algorithm to solve Π_3.

Algorithm 5.1.

Step 1. Order the jobs in non-increasing order of processing times relabel them as $1, \ldots, n$.
Step 2. Assign jobs in batches using First Fit rule as Step 2 of Algorithm 4.1. The obtained batches are $B_1, B_2 \ldots, B_K$ with $T_1 \geq T_2 \geq \ldots T_K$.
Step 3. Reverse the order of the batches and relabel them as $b_1, \ldots, b_K$. Then process $b_1, \ldots, b_K$ one by one. The obtained solution is π_3 and the makespan is $C_{\max}$.

Since jobs have non-identical processing times, we order them by processing time first and then assign them in batches in 5.1. Xia and Tan (2010) show that the performance of First Fit for Bin Packing Problem is $\text{FF} \leq \frac{12}{7}\text{OPT}$, where FF represents the number of bins found by FF and OPT represents the optimal number of bins. In 5.1, Step 1 makes jobs ready for assignment and Step 2 is to assign them in batches. Since in Step 2 we only assign jobs in batches by their sizes, it is equivalent to First Fit for Bin Packing Problem. Therefore, we obtain the following proposition.

Porposition 5.2. *Algorithm 5.1 finds a solution π_3 with $K \leq \frac{12}{7}K^*$.*

Since $K \leq \frac{12}{7}K^* < 2K^*$, we can add $(2K^* - K)$ *empty batches* to the batch list of π_3. An *empty batch* does not cost processing time or machine capacity and it is designed for the convenience of the following analysis. Now the batch list is $\{b_1, \ldots, b_K, b_{K+1}, \ldots, b_{2K^*}\}$, where the latter $(2K^* - K)$ batches are *empty batches*.

Lemma 5.3. *If we order the batches in π_3 and π_3^* in non-increasing order of processing times, then $T_{2k} \leq T_{2k-1} \leq T_k^*$ for all $k = 1, \dots, K^*$.*

Proof. Since batches are ordered in non-increasing order of processing times, $T_1 \geq T_2 \geq \cdots \geq T_{2K^*}$ and $T_1^* \geq T_2^* \geq \cdots \geq T_{K^*}^*$. Find job q which makes $\sum_{j=1}^{q-1} s_j \leq (x-1)D$ and $\sum_{j=1}^{q} s_j > (x-1)D$ in the job list obtained by Step 1 of 5.1. Here x is and arbitrary batch number, *i.e.*, $x \in \{1, \dots, K\}$. Now consider the position of q in π_3^*. If q is assigned in a batch of $b_1^*, b_2^*, \dots, b_x^*$, then at least one job with a longer processing time should be assigned in b_x since $\sum_{j=1}^{q-1} s_j \leq (x-1)D$. By (2.1), $T_x^* \geq t_q$. If q is not assigned in $b_1^*, b_2^*, \dots, b_x^*$, then we also have $T_x^* \geq t_q$. So we obtain

$$T_x^* \geq t_q. \tag{5.1}$$

In π_3, we will show jobs $1, \dots, q$ are all assigned in $b_1, b_2, \dots, b_{2x}$. Consider the worst case where the most batches are needed to accommodate jobs $1, \dots, q$. The worst case appears when any two jobs cannot be combined in a batch, that is, $s_i + s_j > D$ for any $i \neq j$. In the worst case, the number of batches is the same as the case where each job has the same size s and $s > D/2$. Obviously at most $(q-1)$ batches are needed to assign jobs $1, \dots, (q-1)$. Since $\sum_{j=1}^{q-1} s_j \leq (x-1)D$, we have $q - 1 \leq (x-1)D/s < 2(x-1)$. This implies that jobs $1, \dots, (q-1)$ can be assigned in $b_1, \dots, b_{2x-2}$, *i.e.*, q can be assigned in b_{2x-1} because q is the longest job in the remained jobs and has priority to be assigned. So in the worst case $T_{2x-1} = t_q$ and in the other cases, jobs $1, \dots, q$ can be assigned in fewer batches. Therefore,

$$T_{2x-1} \leq t_q. \tag{5.2}$$

By (5.1) and (5.2), $T_{2x-1} \leq T_x^*$. Because batches are ordered in non-increasing order of processing times, we have $T_{2x} \leq T_{2x-1} \leq T_x^*$. Here b_x represents an arbitrary batch and hence $T_{2k} \leq T_{2k-1} \leq T_k^*$. ☐

By Lemma 5.3, we analyze the performance of 5.1 when solving Π_3. The objective function is shown in (2.3), and we first analyze γ/γ^* and then δ/δ^*.

In order to find a bound of γ/γ^*, we consider two cases where $K^* \leq 3$ and $K^* \geq 4$, respectively.

Case 1: $K^* \leq 3$.

Case 1.1: $K^* = 1$. In this case, all the jobs can be assigned in one batch, and $K = 1$. Thus $\gamma = \gamma^* = \max\{t_j : j = 1, \dots, n\}$.

Case 1.2: $K^* = 2$. By proposition 5.2, $K \leq \frac{12}{7} \times 2 = \frac{24}{7}$, *i.e.*, $K \leq 3$. Now consider the worst case where $K = 3$. In the batch list obtained by Step 1, consider job y which makes $\sum_{j=1}^{y-1} s_j \leq D$ and $\sum_{j=1}^{y} s_j > D$ as shown in Figure 2. π_3^* is shown on the left side and π_3 is shown on the right. Since $t_1 \geq t_2 \geq \cdots \geq t_y$, $T_1^* = t_1$ and $T_2^* \geq t_y$ in π_3^*. By contrast, in π_3, $1, \dots, (y-1)$ are all assigned in b_1 since $\sum_{j=1}^{y-1} s_j \leq D$. Therefore, $T_1 = t_1 = T_1^*$, $T_2 = t_y = T_2^*$ and $T_3 \leq t_y = T_2^* \leq T_1^*$. We obtain

$$\frac{\gamma}{\gamma^*} = \frac{T_1 + T_2 + T_3}{T_1^* + T_2^*} = \frac{T_1^* + T_2^* + T_3}{T_1^* + T_2^*} = 1 + \frac{T_3}{T_1^* + T_2^*} \leq 1 + \frac{T_3}{2T_2^*} \leq 1 + \frac{1}{2} = 1.5. \tag{5.3}$$

If $K^* = 2$ and $K = 2$, γ is optimal by the above discussion.

Case 1.3: $K^* = 3$. Now $K \leq \frac{12}{7} \times 3 = \frac{36}{7}$ and we also consider the worst case where $K = 5$. Similar to Case 1.2, find the special job y and we have $T_1 = t_1 = T_1^*$ and $T_2 = T_y = T_2^*$. For the rest batches, we analyze their processing times using (2.4), that is, $r = t_{\min}/t_{\max}$. Because $t_{\min} = \min\{p_j : j = 1, \dots, n\}$ and $t_{\max} = \max\{p_j : j = 1, \dots, n\}$, we have $t_{\min} \leq T_k^* \leq t_{\max}$ for any $k = 1, \dots, K^*$. Thus, $\frac{T_3^*}{T_2^*} \leq \frac{t_{\min}}{T_2^*} \leq \frac{t_{\min}}{t_{\max}} = r$.

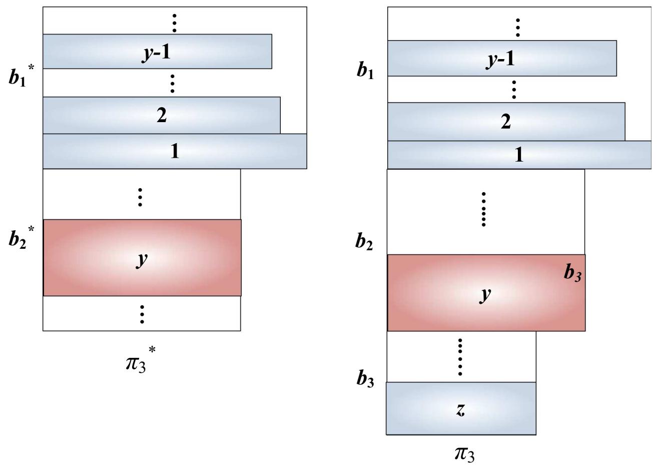

FIGURE 2. The case where $K^* = 2$ and $K = 3$.

So

$$\begin{aligned}\frac{\gamma}{\gamma^*} &= \frac{T_1+T_2+T_3+T_4+T_5}{T_1^*+T_2^*+T_3^*} = \frac{T_1^*+T_2^*+T_3+T_4+T_5}{T_1^*+T_2^*+T_3^*}\\ &\le \frac{T_1^*+T_2^*+T_3+T_4+T_3^*}{T_1^*+T_2^*+T_3^*} = 1+\frac{T_3+T_4}{T_1^*+T_2^*+T_3^*}\\ &\le 1+\frac{2T_2^*}{T_2^*+T_2^*+T_3^*} = 1+\frac{2}{2+\frac{T_3^*}{T_2^*}} \le 1+\frac{2}{2+r}\\ &= 2-\frac{r}{r+2}.\end{aligned} \tag{5.4}$$

Case 2: $K^* \ge 4$. By Lemma 5.3, the batch list in π_3 is $\{b_1, b_2, \ldots, b_{2k-1}, b_{2k}, \ldots, b_{2K^*-1}, b_{2K^*}\}$, which is corresponding to $\{b_1^*, \ldots, b_{k^*}^*, \ldots, b_{K^*}^*\}$ in π_3^* and $T_{2k} \le T_{2k-1} \le T_k$ for $k = 1, \ldots, K^*$. In π_3, batches b_{2k-1}, b_{2k} point to b_k^* in π_3^* for each $k = 1, \ldots, K^*$ if $K = 2K^*$ holds. However, by Proposition 5.2 we find $K \le \frac{12}{7}K^*$, that is, there are less than $2K^*$ batches in π_3. Since two consecutive batches in π_3 points to one batch in π_3^*, the last batch b_K in π_3 does not point to b_K^* in π_3^* but $b_{u^*}^*$ with $u^* < K^*$. Here

$$u^* = \begin{cases} \frac{1}{2}\lfloor \frac{12}{7}K^* \rfloor & \lfloor \frac{12}{7}K^* \rfloor \text{ is even} \\ \frac{1}{2}\left(\lfloor \frac{12}{7}K^* \rfloor + 1\right) & \lfloor \frac{12}{7}K^* \rfloor \text{ is odd.}\end{cases} \tag{5.5}$$

The batches with indexes larger than $\lfloor \frac{12}{7}K^* \rfloor$ are all empty batches, *i.e.*, $T_k = 0$ for $k > \lfloor \frac{12}{7}K^* \rfloor$. By (5.5), we have $\lfloor \frac{12}{7}K^* \rfloor = 2u^*$ or $2u^* - 1$, hence $K = \lfloor \frac{12}{7}K^* \rfloor \le 2u^*$. So

$$\begin{aligned}\frac{\gamma}{\gamma^*} &= \frac{\sum_{k=1}^{K} T_k}{\sum_{k=1}^{K^*} T_k^*} \le \frac{\sum_{k=1}^{u^*}(T_{2k-1}+T_{2k})}{\sum_{k=1}^{K^*} T_k^*}\\ &\le \frac{2\sum_{k=1}^{u^*} T_k^*}{\sum_{k=1}^{u^*} T_k^* + \sum_{k=u^*+1}^{K^*} T_k^*} = \frac{2}{1+\frac{T_{u^*+1}^*+\cdots+T_{K^*}^*}{T_1^*+\cdots+T_{u^*}^*}}\\ &\le \frac{2}{1+\frac{(K^*-u^*)T_{K^*}^*}{u^*T_1^*}} \le \frac{2}{1+\frac{K^*-u^*}{u^*}\times\frac{t_{\min}}{t_{\max}}}\\ &= \frac{2}{1+\left(\frac{K^*}{u^*}-1\right)r}.\end{aligned} \tag{5.6}$$

TABLE 1. Bounds of γ/γ^* for $K^* = 4, 5, 6, 7$.

K^*	4	5	6	7
$12K^*/7$	$\frac{48}{7}$	$\frac{60}{7}$	$\frac{72}{7}$	$\frac{84}{7}$
$\lfloor 12K^*/7 \rfloor$	6	8	10	12
u^*	3	4	5	6
K^*/u^*	$\frac{4}{3}$	$\frac{5}{4}$	$\frac{6}{5}$	$\frac{7}{6}$
γ/γ^*	$2 - \frac{2r}{r+3}$	$2 - \frac{2r}{r+4}$	$2 - \frac{2r}{r+5}$	$2 - \frac{2r}{r+6}$

TABLE 2. Bounds of γ/γ^* for $K^* = 7m + i$.

K^*	$7m+1$	$7m+2$	$7m+3$	$7m+4$	$7m+5$	$7m+6$	$7m+7$
$\frac{12}{7}K^*$	$12m + \frac{12}{7}$	$12m + \frac{24}{7}$	$12m + \frac{36}{7}$	$12m + \frac{48}{7}$	$12m + \frac{60}{7}$	$12m + \frac{72}{7}$	12m+12
$\lfloor \frac{12}{7}K^* \rfloor$	$12m+1$	$12m+3$	$12m+5$	$12m+6$	$12m+8$	$12m+10$	$12m+12$
u^*	$6m+1$	$6m+2$	$6m+3$	$6m+3$	$6m+4$	$6m+5$	$6m+6$
K^*/u^*	$\frac{7m+1}{6m+1}$	$\frac{7m+2}{6m+1}$	$\frac{7m+3}{6m+3}$	$\frac{7m+4}{6m+3}$	$\frac{7m+5}{6m+4}$	$\frac{7m+6}{6m+5}$	$\frac{7}{6}$
γ/γ^*	$2 - \frac{2r}{r+6+\frac{1}{m}}$	$2 - \frac{2r}{r+6+\frac{2}{m}}$	$2 - \frac{2r}{r+6+\frac{3}{m}}$	$2 - \frac{2r}{r+6+\frac{4}{m}}$	$2 - \frac{2r}{r+6+\frac{5}{m}}$	$2 - \frac{2r}{r+6+\frac{6}{m}}$	$2 - \frac{2r}{r+6}$

If $K^* = 4$, then $\lfloor \frac{12}{7}K^* \rfloor = \lfloor \frac{48}{7} \rfloor = 6$. By (5.5) we find $u^* = 3$ and by (5.6),

$$\frac{\gamma}{\gamma^*} \leq \frac{2}{1 + \left(\frac{4}{3} - 1\right) r} = 2 - \frac{2r}{r+3}. \tag{5.7}$$

In the similar way, we obtain the bound of γ/γ^* as shown in Table 1 for $K^* \in \{4, 5, 6, 7\}$.

Here $0 < r \leq 1$. If $r = 1$, $t_{\min} = t_{\max}$, which implies that all the jobs have the same processing time. Since for $K^* \in \{4, 5, 6, 7\}$, the difference between K^* and u^* is one unit, and $2 - \frac{2r}{r+u^*} = 2 - \frac{2}{1+\frac{u^*}{r}}$ is an increasing function of u^*, the worst ratio of γ^* appears when $K^* = 7$.

Lemma 5.4. *When there are no more than seven batches in π_3^*, $\gamma/\gamma^* \leq 12/(r+6)$.*

Now we consider the case where $K^* \geq 8$. In this case we can find two positive integers m and i to make $K^* = 7m + i$ where $1 \leq i \leq 7$. As in the analysis of Case 1.4, the results of γ/γ^* are shown in Table 2. Line 1 shows the value of K^*. Lines 2 and 3 show the value of $\frac{12}{7}K^*$ and $\lfloor \frac{12}{7}K^* \rfloor$. Line 4 shows the value of u^* which is obtained from (5.5) and line 5 shows the ratio of K^*/u^*. Line 6 shows the bound of γ/γ^* which can be obtained from (5.6).

For a given m, the bounds of γ/γ^* are shown in Line 6 of Table 2 and we find that the worst result is $2 - \frac{2r}{r+6+\frac{i}{m}}$ which appears when $i = 6$, *i.e.*, $K^* = 7m + 6$. Since $2 - \frac{2r}{r+6+\frac{6}{m}}$ is a decreasing function of m, the worst bound appears when $m = 1$ and $i = 6$, that is $K^* = 13$. So when $K^* \geq 8$,

$$\frac{\gamma}{\gamma^*} \leq \left.\frac{\gamma}{\gamma^*}\right|_{K^*=13} = 2 - \frac{2r}{r+6+6} = \frac{24}{r+12}. \tag{5.8}$$

Lemma 5.5. *When there are more than seven batches in π_3^*, $\gamma/\gamma^* \leq 24/(r+12)$.*

By Lemmas 5.4 and 5.5, we only need to compare $12/(r+6)$ with $24/(r+12)$ and then we can find the upper bound of γ/γ^* for all K^*. Since

$$\frac{12}{r+6} - \frac{24}{r+12} = -\frac{12r}{(r+6)(r+12)} < 0, \tag{5.9}$$

we have the following theorem.

Theorem 5.6. *For any instance of* Π_3, $\gamma/\gamma^* \leq 24/(r+12)$.

Now we investigate the ratio of δ/δ^*, that is, $\sum_{k=1}^{K} k^\alpha T_k / \sum_{k=1}^{K^*} k^\alpha T_k^*$. By Lemma 3.3, we find in an optimal solution π_3^*, batches are ordered in non-decreasing order of processing times. That is $T_1^* \leq T_2^* \leq \cdots \leq T_{K^*}^*$. Actually, since we have added $(2K^* - K)$ empty batches in our solution π_3 and batches are ordered in non-decreasing order, $T_1 = \cdots = T_{2K^*-K} = 0$. That is, $b_1, \ldots, b_{2K^*-K}$ are all empty batches. By Lemma 5.3, for any $k = 1, \ldots, K^*$

$$T_{2k-1} \leq T_{2k} \leq T_k^*. \tag{5.10}$$

Theorem 5.7. *For any instance of* Π_3, $\delta/\delta^* \leq 1 + 2^\alpha$, *where* α *is the learning index and* $\alpha < 0$.

Proof. First we define the following function

$$f_{(k)} = \frac{(2k-1)^\alpha + (2k)^\alpha}{k^\alpha}. \tag{5.11}$$

We have

$$f_{(k)} = \frac{(2k-1)^\alpha + (2k)^\alpha}{k^\alpha} = \left(2 - \frac{1}{k}\right)^\alpha + 2^\alpha. \tag{5.12}$$

Since $\alpha < 0$,

$$f'_{(k)} = \frac{\alpha\left(2 - \frac{1}{k}\right)^{\alpha-1}}{k^2} < 0. \tag{5.13}$$

This implies that $f_{(k)}$ is a decreasing function of k., *i.e.*, $f_{(k)} \leq f_{(1)}$. So

$$\begin{aligned}
\frac{\delta}{\delta^*} &= \frac{1^\alpha T_1 + 2^\alpha T_2 + \cdots + (2K^*-1)^\alpha T_{2K^*-1} + (2K^*)^\alpha T_{2K^*}}{1^\alpha T_1^* + \cdots + (K^*)^\alpha T_{K^*}^*} \\
&= \frac{\sum_{k=1}^{K^*}((2k-1)^\alpha T_{2k-1} + (2k)^\alpha T_{2k})}{\sum_{k=1}^{K^*} k^\alpha T_k^*} \\
&\leq \frac{\sum_{k=1}^{K^*}((2k-1)^\alpha + (2k)^\alpha) T_k^*}{\sum_{k=1}^{K^*} k^\alpha T_k^*} \\
&\leq \frac{\sum_{k=1}^{K^*} f_{(k)} k^\alpha T_k^*}{\sum_{k=1}^{K^*} k^\alpha T_k^*} \\
&\leq \frac{\sum_{k=1}^{K^*} f_{(1)} k^\alpha T_k^*}{\sum_{k=1}^{K^*} k^\alpha T_k^*} = f_{(1)} = 1 + 2^\alpha.
\end{aligned} \tag{5.14}$$

Theorem 5.7 follows. □

By Theorems 5.6 and 5.7, the performance guarantee of 5.1 is $\max\{\frac{24}{r+12}, 1+2^\alpha\}$. Since

$$\frac{24}{r+12} - (1+2^\alpha) = \frac{12-r}{12+r} - 2^\alpha. \tag{5.15}$$

So

$$\frac{C_{\max}}{C_{\max}^*} \leq \max\left\{\frac{24}{r+12}, 1+2^\alpha\right\} = \begin{cases} \frac{24}{r+12} & \alpha \leq \log_2 \frac{12-r}{12+r} \\ 1+2^\alpha & \alpha > \log_2 \frac{12-r}{12+r} \end{cases}. \tag{5.16}$$

Observe (5.16) and we have $\frac{C_{\max}}{C_{\max}^*} < 2$ regardless of the value.

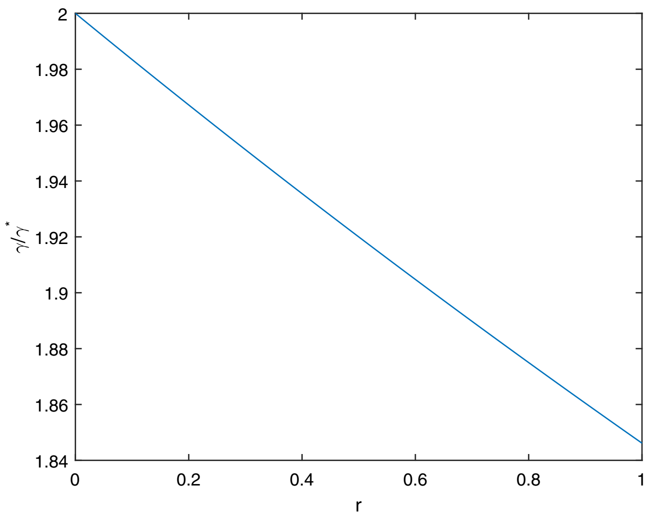

FIGURE 3. Performance influenced by r.

Theorem 5.8. *When solving* Π_3, *Algorithm* 5.1 *can find a solution with* $\frac{C_{\max}}{C^*_{\max}} \le \max\left\{\frac{24}{r+12}, 1+2^{\alpha}\right\} < 2$ *in* $O(n^2)$ *time, where* $C^*_{\max}$ *is the optimal makespan.*

Now we analyze the evolution of performance of 5.1 caused by K^*, r and α respectively, which are key parameters of the problem. First, the worst result of γ/γ^* is $\frac{24}{r+12}$ which appears when $K^* = 13$ as shown in Lemma 5.5. When $K^* = 1$, Algorithm 5.1 is optimal. When $K^* > 13$, the performance guarantee is better than $\frac{24}{r+12}$. This implies that a better demand can bring a better production efficiency. In order to improve the production system, marketing should be improved first. When more products are needed, the makespan can be reduced even better. In view of the problem scale, the operations department of a manufacturing company is not isolated from the other departments but closely related to the other departments, especially the marketing department.

Second, by Theorem 5.6 the influence of r on the production efficiency is shown in Figure 3. The performance guarantee of 5.1 is a decreasing function of r, which is the ratio of $t_{\min}/t_{\max}$. That is, the performance becomes better when r increases. A larger r means more differences between products, *i.e.*, the customers have different requirements on the processing times. Therefore, diversity of products makes it difficult to optimize the production system. For the decision makers, it is important to find a trade-off between the diversity of products and production efficiency. Diversity of products can improve customer's satisfaction and custom-made products are even better, however, the diversity makes it difficult to improve makespan simultaneously. On the other aspect, we find that the performance of 4.1 is better than 5.1 and the reason is that jobs have the same size in Π_2 but different sizes in Π_3. It also implies the diversity of products brings difficulty to the production system.

Third, the learning effects have an obvious influence on the performance of 5.1 as shown in Figure 4. The performance guarantee is an increasing function of α, that is, Algorithm 5.1 performs worse when α becomes larger. A larger α means worse learning effects, *i.e.*, the workers cannot improve their abilities effectively. Workers are the leading factor of learning effects and thus, better workers are necessary for a better production system. From Figure 4, we find when $\alpha = -1$ the learning effects are the best and $\delta/\delta^* \le 1.5$. However, when $\alpha \to 0$ there is no learning effect and $\delta/\delta^* \le 2$. For the decision makers, it is important to improve the learning effects and design effective rules to motivate workers to enhance their skills.

Finally, by Theorem 5.8 we present the evolution of the performance by the change of r and α as shown in Figure 5. The worst result appears when $r \to 0$, *i.e.*, the differences between jobs are sufficiently large or

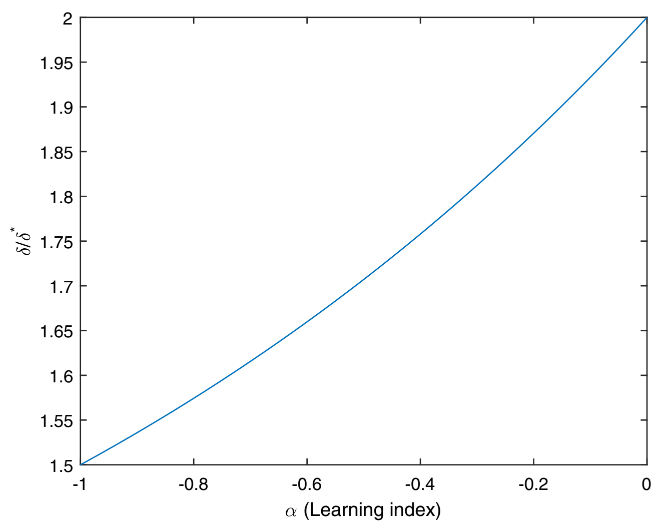

Figure 4. Performance influenced by learning index.

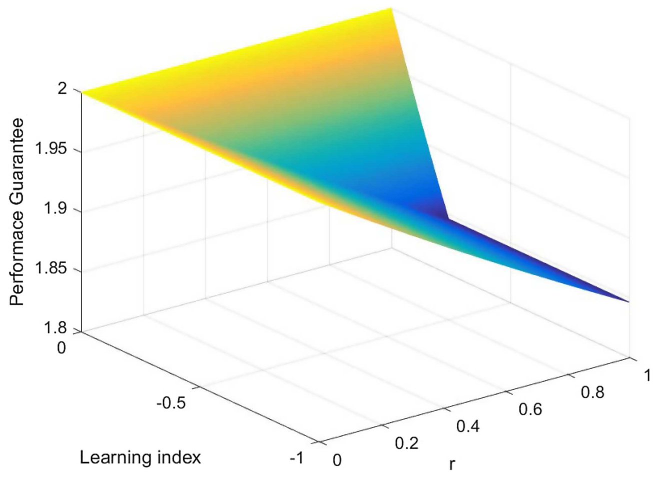

Figure 5. Evolution of performance guarantee by the change of r and α.

$\alpha = 0$, *i.e.*, there is no learning effect for workers. For decision makers, the diversity of products and learning effects need to be considered at the same time. As shown in Figure 5, when $r \to 1$ and $\alpha \to -1$, Algorithm 5.1 performs well. The production efficiency can be guaranteed if there are effective learning effects and the diversity of products is designed well.

6. Computational experiments

In order to show the performance of our proposed algorithm, we conduct experiments. Since problem Π_3 is the general model and is most difficult to solve, we generate random instances of Π_3. The parameters are set as follows. Let $D = 50$, that is, the machine capacity is fixed. The sizes of jobs are classified into three levels, $S1$, $S2$ and $S3$ where the size of a job obeys uniform distribution on the interval (1, 10), (1, 25) and (1, 50). Parameter M is also classified in 3 levels where $M1 = 0.1$, $M2 = 0.5$ and $M3 = 0.9$ respectively. Parameter r

TABLE 3. Experimental results of $C_{\max}/C^*_{\max}$.

n	100			500			1000		
α	-0.9	-0.5	-0.1	-0.9	-0.5	-0.1	-0.9	-0.5	-0.1
$S1M1r1$	1.201	1.246	1.258	1.265	1.270	1.301	1.275	1.284	1.288
$S1M1r2$	1.225	1.240	1.254	1.245	1.269	1.290	1.265	1.281	1.291
$S1M1r3$	1.267	1.278	1.296	1.278	1.287	1.295	1.283	1.290	1.321
$S1M2r1$	1.270	1.275	1.279	1.281	1.298	1.301	1.269	1.280	1.289
$S1M2r2$	1.261	1.270	1.275	1.280	1.283	1.289	1.281	1.283	1.295
$S1M2r3$	1.265	1.271	1.275	1.275	1.279	1.280	1.279	1.286	1.289
$S1M3r1$	1.263	1.275	1.289	1.268	1.271	1.270	1.269	1.271	1.270
$S1M3r2$	1.271	1.289	1.286	1.274	1.290	1.294	1.281	1.281	1.285
$S1M3r3$	1.261	1.274	1.290	1.271	1.281	1.291	1.285	1.281	1.288
$S2M1r1$	1.291	1.294	1.299	1.298	1.291	1.291	1.301	1.304	1.311
$S2M1r2$	1.290	1.291	1.298	1.298	1.290	1.314	1.291	1.299	1.347
$S2M1r3$	1.310	1.303	1.318	1.311	1.320	1.300	1.315	1.323	1.318
$S2M2r1$	1.301	1.327	1.321	1.332	1.323	1.317	1.324	1.341	1.321
$S2M2r2$	1.318	1.325	1.328	1.341	1.325	1.322	1.300	1.336	1.328
$S2M2r3$	1.321	1.325	1.322	1.330	1.341	1.345	1.331	1.335	1.345
$S2M3r1$	1.320	1.324	1.326	1.331	1.328	1.332	1.335	1.341	1.328
$S2M3r2$	1.340	1.338	1.338	1.331	1.332	1.300	1.341	1.340	1.351
$S2M3r3$	1.356	1.367	1.398	1.320	1.371	1.384	1.398	1.381	1.391
$S3M1r1$	1.381	1.361	1.388	1.390	1.458	1.401	1.411	1.461	1.410
$S3M1r3$	1.440	1.451	1.500	1.419	1.443	1.429	1.432	1.480	1.481
$S3M2r1$	1.481	1.491	1.430	1.541	1.436	1.438	1.447	1.451	1.472
$S3M2r2$	1.581	1.498	1.4485	1.473	1.491	1.468	1.500	1.428	1.439
$S3M2r3$	1.480	1.529	1.521	1.471	1.520	1.503	1.532	1.528	1.537
$S3M3r1$	1.583	1.479	1.581	1.561	1.520	1.473	1.526	1.548	1.623
$S3M3r2$	1.535	1.562	1.534	1.526	1.545	1.594	1.530	1.537	1.542
$S3M3r3$	1.541	1.524	1.538	1.543	1.500	1.547	1.628	1.523	1.526

indicates the difference between processing time of jobs and three levels of r are tested where $r1 = 1$, $r2 = 5$ and $r3 = 10$. So an instance is denoted by $SaMbrc$, where $a, b, c = 1, 2, 3$. For example, instance $S2M1r3$ represents an instance where job sizes obey uniform distribution on (1, 25), $M = 0.1$ and $r = 10$. We run each instance 20 times and show the average value of $C_{\max}/C^*_{\max}$, where $C^*_{\max}$ is represented by a lower bound since Π_3 is NP-hard in the strong sense. Obviously the actual results are better than the presented results since the lower bound is better than $C^*_{\max}$. Let R represent the worst case ratio of 5.1.

The results are presented in Table 3, where different scales of instances are tested. Line 1 shows the scale of the instances where $n = 100, 500$ and 1000 respectively. In each scale, we consider three levels of learning effects where $\alpha = -0.9$, -0.5 and -0.1 respectively as shown in Line 2. Then we present the worst-case ratio of our algorithm in all 27 instances. From the results we find the learning effects influence the results obviously. In most cases, the performance becomes better when α becomes larger, that is, there are better learning effects. The best result is 1.201, which appears in the instance $S1M1r1$. The worst result is 1.628, which appears in the instance $S3M3r3$. Since the optimal $C^*_{\max}$ is replaced by a lower bound, the actual results are better than those in Table 3. We find that even in the largest instance, the worst case ratio is strictly less than 2, which demonstrates the effectiveness of our proposed algorithm.

7. Conclusions

In this paper, we investigate the problem of scheduling a single batch-processing machine with learning effects. Motivated by applications in real industries such as porcelain companies, electroplating companies, food making companies and metal working companies, we propose batch-position-based learning effects. Jobs are assigned in batches and then batches are processed one by one. Workers become more and more skillful after they have processed batches repeatedly, which makes the actual processing time of a batch less than its normal processing time. We propose problems of minimizing makespan with batch-position-based learning effects and provide effective algorithms. Three problems are considered where the second and third problems are both NP-hard in the strong sense. We provide an algorithm for the first problem and approximation algorithms for the NP-hard problems. The time complexity and performance guarantees are analyzed and the evolution of the performance is shown.

For future research, many interesting problems keep open and deserve investigation. First, scheduling multiple batch-processing machines with learning effects is an interesting direction. Since the general problem of scheduling a single batch-processing machine is NP-hard in the strong sense, the problem of scheduling multiple batch-processing machines is also NP-hard in the strong sense. The problems are more difficult to solve and approximation algorithms need further investigation. Second, the supply chain scheduling problem with learning effects is an interesting direction. In this paper, we only consider the production part, however, the inventory and distribution parts are also important for manufacturers for minimizing operational cost and serving customers. As in the production part, learning effects also exist in the inventory and distribution parts and need investigation. Therefore, the supply chain scheduling problem is more complex and the algorithms deserve more study. Finally, the management methods are also interesting for research. Since workers become skillful gradually in the supply chain system, effective motivating methods deserve investigation. Such a method can motivate workers to improve their learning effects, which is important to the operations of the manufacturers. The derivation of the management methods relies on further research on learning effects of workers and optimization methodology on supply chain.

Acknowledgements. This work is supported by Anhui soft science project under grant number 1607a0202031, Humanities and Social Sciences in Anhui Universities under Grant number SK2016A0526 and research project of innovation and development of Social Sciences in Anhui under Grant number 2017CXF083.

References

[1] S. Akiyoshi, V.S. Natalia and A.S. Vitaly, Application of submodular optimization to single machine scheduling with controllable processing times subject to release dates and deadlines. *INFORMS J. Comput.* **28** (2016) 148–161.

[2] D. Biskup, Single-machine scheduling with learning considerations. *Eur. J. Oper. Res.* **115** (1999) 173–178.

[3] T.C.E. Cheng, C.T. Ng, J.J. Yuan and Z.H. Liu, Single machine parallel batch scheduling subject to precedence constraints. *Nav. Res. Logist.* **51** (2004) 949–958.

[4] B.Y. Cheng, J.Y.T. Leung, K. Li and S.L. Yang, Single batch machine scheduling with deliveries. *Nav. Res. Logist.* **62** (2015) 470–482.

[5] J.R. DeJong, The effects of increasing skill on cycle time and its consequences for time standards. *Economics* **1** (1957) 51–60.

[6] G. Dosa, R. Li, X. Han and Z. Tuza, Tight absolute bound for first fit decreasing bin packing: $\text{FFD}(L) \leq 11/9\text{OPT}(L) + 6/9$. *Theor. Comput. Sci.* **510** (2013) 13–61.

[7] D. Giglio, Optimal control strategies for single-machine family scheduling with sequence-dependent batch setup and controllable processing times. *J. Sched.* **18** (2015) 525–543.

[8] N.P.A. Hidayat, A. Cakravastia, T.M.A.A. Samadhi and A.H. Halim, A batch scheduling model for m heterogeneous batch processor. *Int. J. Prod. Res.* **54** (2016) 1170–1185,

[9] X. Huang, M.Z. Wang and P. Ji, Parallel machines scheduling with deteriorating and learning effects. *Optim. Lett.* **8** (2014) 493–500.

[10] M. Ji, D. Yao, Q. Yang and T.C.E. Cheng, Machine scheduling with DeJongs learning effect. *Comput. Ind. Eng.* **80** (2014) 195–200.

[11] Z. Jiang, F. Chen and H. Kang, Single-machine scheduling problems with actual time-dependent and job-dependent learning effect. *Eur. J. Oper. Res.* **115** (2013) 173–178.

[12] C. Koulamas, A note on single-machine scheduling with job-dependent learning effects. *Eur. J. Oper. Res.* **207** (2010) 1142–1143.

[13] P.J. Lai and W.C. Lee, Single-machine scheduling with general sum-of-processing-time-based and position-based learning effects. *OMEGA Int. J. Manage. Sci.* **39** (2011) 467–471.

[14] W.C. Lee, Scheduling with general position-based learning curves. *Inf. Sci.* **181** (2011) 5515–5522.

[15] W.C. Lee, C.C. Wu and P.H. Hsu, A single-machine learning effect scheduling problem with release times. *OMEGA Int. J. Manage. Sci.* **28** (2010) 3–11.

[16] Y. Li, Combined scheduling algorithm for re-entrant batch-processing machines in semiconductor wafer manufacturing. *Int. J. Prod. Res.* **53** (2015) 1866–1879.

[17] M. Li and N.C. Petruzzi, Demand uncertainty reduction in decentralized supply chains. *Prod. Oper. Manage.* **26** (2017) 156–161.

[18] M. Liu, Parallel-machine scheduling with past-sequence-dependent delivery times and learning effect. *Appl. Math. Model.* **37** (2013) 9630–9633.

[19] L.L. Liu, C.T. Ng and T.C.E. Cheng, Scheduling jobs with release dates on parallel batch processing machines. *Discrete Appl. Math.* **157** (2009) 1825–1830.

[20] D. Okolowski and S. Gawiejnowicz, Exact and heuristic algorithms for parallel-machine scheduling with DeJongs learning effect. *Comput. Ind. Eng.* **59** (2010) 272–279.

[21] M.L. Pinedo, Scheduling: Theory, Algorithms, and Systems. Springer (2008).

[22] J. Qian and G. Steiner, Fast algorithms for scheduling with learning effects and time-dependent processing times on a single machine. *Eur. J. Oper. Res.* **225** (2013) 547–551.

[23] Y. Tan and J. Carrillo, Strategic analysis of the agency model for digital goods. *Prod. Oper. Manage.* **24** (2017) 724–741.

[24] Y. Tan, J. Carrillo and H.K. Cheng, The agency model for digital goods. *Decis. Sci.* **47** (2016) 628–660.

[25] L.X. Tang, G.S. Wang and Z.L. Chen, Integrated charge batching and casting width selection at baosteel. *Oper. Res.* **62** (2014) 772–787.

[26] T.T. Tran, A. Araujo and J.C. Beck, Decomposition methods for the parallel machine scheduling problem with setups. *INFORMS J. Comput.* **28** (2016) 83–95.

[27] J.Q. Wang and J.Y.T. Leung, Scheduling jobs with equal-processing-time on parallel machines with non-identical capacities to minimize makespan. *Int. J. Prod. Econ.* **156** (2014) 325–331.

[28] H. Wiebke, K.G. Felix, M.H. Rolf and L.E. Marco, Integrated sequencing and scheduling in coil coating. *Manage. Sci.* **57** (2011) 647–666.

[29] D.L. Yang and W.H. Kuo, A single-machine scheduling problem with learning effects in intermittent batch production. *Comput. Ind. Eng.* **57** (2009) 762–765.