Group scheduling with non-periodical maintenance and deteriorating effects

Haiyan Xu, Xiaoping Li, *Senior Member, IEEE,* Rubén Ruiz and Haihong Zhu

***Abstract*—In this paper, we consider single machine group scheduling with non-periodical maintenance and deteriorating effects. Non-periodical maintenance, which has unfixed maintaining interval or the number of jobs in each group is unfixed, results in a variable number of groups. Deteriorating effects lead to longer processing times of which the deterioration index depends on job grouping. This problem is of significance in different production settings and is much more difficult than and general that other simpler single machine group scheduling problems. Making use of historical processing times, we construct the actual processing time model for jobs. We prove that the problem under study is NP-hard. By transforming the optimization objective, properties are discovered and two batch-based heuristics are presented for small size problems. To further improve the effectiveness for large size problems, an iterated greedy algorithm is proposed being its main advantages simplicity and effectiveness. The proposed methods are evaluated over a large number of random instances with calibrated parameters and components. Comprehensive computational and statistical analyses demonstrate the superiority of the methods proposed over adapted existing approaches.**

Keywords*—Group scheduling, Single machine, Deteriorating effects, Non-periodical maintenance.

I. Introduction

Traditionally, processing times of tasks or jobs are known in advance and assumed to be deterministic in scheduling problems [1]–[3]. However, some factors make the processing times of tasks or jobs stochastic: (i) Workers' skills are usually improved by repetitively processing similar activities over and over. This phenomenon is known as the "learning effect" [4], which results in actual processing times that are shorter than normal ones. (ii) On the other hand, actual processing times of jobs can be deteriorated because of frayed grinding wheels, interruption of learning and changeovers between different batches of product amongst other reasons. This phenomenon is known as "deteriorating effect" [5], which results in processing times that are longer than normal ones. Furthermore, these factors are usually intertwined which make it much hard to estimate them in practical production manufacturing environments [6].

Maintenances are closely related to both learning and deteriorating effects. Though maintenance usually improves the efficiency of the machines, they deteriorate jobs' processing times due to the interruption of the learning effect. In addition, more groups or more maintenances result in fewer deteriorating effects or shorter actual processing times but they lead to longer maintenance durations. Therefore, it is desirable to strike a balance between actual processing times and maintenance times. Maintenances are common in practical productions. For example, the missile radome manufacturing process is complex, hard, expensive, and precise. The generatrix is a continuous smooth curve which is placed on grinding machines and processed by grinding wheels. Maintaining grinding machines is necessary to keep them in good working condition after processing some products. Moreover, there are near 100 machines in an aircraft plant. Each machine is maintained every month on average and the maintenance time ranges from several hours to several days.

In this paper, we consider a number of jobs being processed on a single machine with non-periodical maintenance and deteriorating effects. This problem is different from the group scheduling problems with a fixed number of groups and given jobs in each group [7]–[11]. Usually learning effects are human oriented while deteriorating effects are manufacturing process oriented. The former has far more flexibility in terms of control than the latter. It is hard to maintain a machine with fixed maintenance intervals for unpredicted requirements of jobs. Different numbers of groups and varying processing times make maintenances non-preventive, i.e, maintenances of machines are usually non-periodical (maintenance intervals or the number of jobs in each group are unfixed) in practice. The actual processing times of different jobs vary when they are placed in different positions of a group because of deteriorating effects. For jobs on each machine, different grouping manners lead to distinct maintenance times and maintenance durations. They interact on each other which make the problem under study much complex.

The main contributions of this paper are: (i) The considered problem is modelled mathematically and its NP-hardness is proven. (ii) Two batch-based heuristics are proposed for the considered problem for small size instances and an iterated greedy algorithm is presented for large size ones. (iii) The use of time series analysis techniques on historical real-time data to create a time-dependent deteriorating effect.

The rest of this paper is organized as follows. Section II summarizes related work. The considered problem is modelled mathematically and its properties are analyzed in Section III.

Haiyan Xu, Xiaoping Li and Haihong Zhu are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China, and also with Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, Nanjing, 211189, China. Haiyan Xu is also with the Department of Public Basic Course, Jiangsu Key Laboratory of Data Science & Smart Software, JinLing Institute of Technology, Nanjing, China (e-mail: 455457802@qq.com, xpli@seu.edu.cn, haih.zhu@seu.edu.cn).

Rubén Ruiz is with Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edifico 8G, Acc. B. Universitat Politècnica de València, Camino de Vera s/n, 46021, València, Spain (e-mail: rruiz@eio.upv.es).

Section IV investigates two heuristics and their evaluation. An iterated greedy algorithm is proposed in Section V. The iterated greedy algorithm is evaluated in Section VI, followed by conclusions in Section VII.

II. Related work

During the past decade, much attention has been paid to scheduling problems concerning learning and/or deteriorating effects. Basically, there are two types of learning and/or deteriorating effects [12]: position-based and time-based. The position-based ones [13] [14] depend on the number of processed jobs while the time-based ones take into account the processing time of all processed jobs so far. In addition, most single-machine group scheduling problems with learning and/or deteriorating effects assume a fixed number of groups and also a fixed number of jobs in each group, i.e., periodical maintenance.

Some single-machine group scheduling problems with specific learning and/or deteriorating effect functions have been proved to be solvable in polynomial time. Yang et al. [7], Yang [8] considered single machine group scheduling problems involving deteriorating and/or learning effects. They assumed that the group setup time or the actual processing time function of a job was time-based and/or position-based learning and/or deteriorating. They proved that makespan and the total completion time minimization problems could be optimally solved in polynomial time under some given conditions. Yin et al. [9], Yeh et al. [15] and He and Sun [10] studied single machine scheduling problems with deteriorating and learning effects. They showed that the addressed problems remained polynomially solvable for some objectives (makespan, total completion time and the total weighted completion time). Huang et al. [11] considered single machine group scheduling problems with learning and/or deteriorating effects in which actual processing times depend on allocated resources. They proved that the problems for some objectives were polynomially solvable under some conditions. Wu et al. [16] proposed a generalized model with past-sequence-dependent learning and deteriorating effect. The effects were assumed to be dependent on both the sum of processing times and the scheduled position. They investigated some single-machine problems with various objectives: makespan, total completion time, weighted completion time and maximum lateness. In addition, they proved the problems to be solvable in polynomial time under certain conditions.

Most single-machine group scheduling problems with learning and/or deteriorating effects have not been proved to be solvable in polynomial time. Therefore, some exact and heuristic algorithms have been proposed. Ji et al. [17] considered several single machine scheduling problems with deteriorating effects, slack due date assignment, resource allocation and a rate-modifying activity. The actual processing time of a job depends on both its position in a processing sequence and the amount of resource allocated. Keshavarz et al. [18] investigated a single machine sequence-dependent group scheduling problem to minimize the total weighted earliness and tardiness respectively. Xu et al. [19] showed that the single machine group scheduling problem with deterioration effects is solvable in polynomial time. Rustogi et al. [20] proposed several heuristics for single machine group schedule problems with generalized positional deterioration effects to minimize makespan. They considered machine maintenance between adjacent groups. The deterioration factor of a position in a group is constant, which is different from the problem under study in this paper.

Deterioration results in higher production costs and lower product quality which can be increased by interrupting the current manufacturing process and carrying out maintenance over the machine. In other words, the efficiency of the machine can be restored or partially restored by maintenance operations. Generally, there are two types of maintenance: preventive and non-preventive. Preventive maintenance refers to maintaining the machine after a given number of jobs or after a given number of time periods (periodical maintenance) [21]. At present, periodical maintenance is usually considered in single machine group scheduling problems with learning and/or deteriorating effects. Pan et at. [22], and Liu et al. [23] considered the problem with both learning and/or deteriorating effects and periodic maintenance to minimize makespan or the number of tardy jobs respectively. Iranpoor et al. [24] studied a single machine scheduling problem with a common due date for all jobs, sequence dependent setup times and a rate-modifying activity. The rate-modifying activity determines whether to bring the machine completely or partially back to normal after perfect or imperfect maintenance. The sum of the earliness cost, the tardiness cost and the due date related cost is minimized. They transformed the problem into a time dependent traveling salesman problem. A B&B procedure was presented for small problems and two robust heuristics were proposed for larger problems. Though the single machine group scheduling problem considered by Zhang et al. [25] is similar to the considered problem in this paper to some extent, they are different in two crucial aspects: (1) The deterioration index of the former is constant in every group and each position of every group while it is distinct for the latter. (2) Deteriorating effects of the former depend on jobs while those of the latter depend on the machine.

Because no job can be interrupted while it is being processed (i.e., no preemption is allowed), it is hard to ensure that jobs are finished before maintenance. In other words, non-periodical maintenance is more practical. To the best of our knowledge, there is no existing work considering both non-periodical maintenance (non-fixed number of groups and non-determined jobs in each group) and deteriorating effects. In this paper, we consider a single machine group scheduling problem with non-periodical maintenance and deteriorating effects with the objective of minimizing makespan.

III. Problem Description and Properties

In this paper, we consider an n-job single-machine scheduling problem. There is a job set $\mathbb{J} = \{J_1, J_2, \cdots, J_n\}$, where all jobs are available at time zero and independent of each other, i.e., there are no precedence constraints. Initially, the machine is assumed to be in a perfect state and its processing capability deteriorates after processing jobs. After maintenance, the capability of the machine can be partially or completely restored.

Jobs cannot be processed on the machine during maintenance. Processing times of jobs in each group have deteriorating effects. The n jobs are partitioned into m $(1 \le m \le n)$ groups or $m-1$ maintenances. How many groups are partitioned and how to allocate the n jobs to each group is crucial in getting an optimal makespan for the considered problem. We first list the notations to be used in Table I.

TABLE I: Notations employed in the paper

G_i	:	The ith group, $i = 1, 2, ..., m$.
n_i	:	Number of jobs in G_i, i.e., $\sum_{i=1}^{m} n_i = n$.
J_j	:	The job j.
$J_{i,[j]}$	:	The j^{th} job scheduled inside G_i.
s_i	:	Maintenance time between the processing of G_i and G_{i+1} $(i = 1, ..., m-1)$.
$p_{[k]}^{(i)}$	:	Actual processing time of the k^{th} job in G_i,$k = 1, 2, ..., n_i$.
$p_{i,j}^{r}$	:	Actual processing time of J_j if it is scheduled at the r^{th} position of G_i.
q_j	:	Normal processing time of J_j.
q_i^k	:	Normal processing time of the k^{th} job in G_i.
$f_{i,[j]}$	:	Completion time of $J_{i,[j]}$.
C_i	:	Completion time of G_i, i.e., $C_i = f_{i,[n_i]}$.
$C_{\max}$	:	Makespan of a schedule, i.e., $C_{\max} = C_m$.
b_i	:	Deterioration index of G_i.

To improve the robustness of schedules, Liu et al. [26] used a fuzzy variable with a continuous membership function and finite expected value to estimate the actual processing time of a job. Liu et al. [27] studied a loss-averse news-vendor problem with random yield. For single-machine scheduling problems, Kuo and Yang [28] first introduced the sum-of-processing-time-based deteriorating effect with the function

$$p_{j,r} = q_j(1 + \sum_{k=1}^{r-1} p_{[k]})^a, \quad j = 1, 2, \ldots, n. \qquad (1)$$

where q_j is the normal processing time of J_j. $p_{j,r}$ is the actual processing time of J_j if it is scheduled in position r in a sequence, $p_{[k]}$ is the actual processing time of the job located at the k^{th} position and a is a constant learning index. This model and its variants consider the processed jobs before J_j. However, similar products or items are usually processed on the same machine in practice which implies that deteriorating effects can be predicted by historical real-time data, i.e., some functions between the normal and actual processing times could be constructed based on a large amount of data analyzed by the time series analysis method. In this paper, we construct the actual processing time model with deteriorating effects using the time series analysis method which generally includes the trend term, the periodic term and the stochastic term. Only the trend term is usually employed. The fitting curve of the trend term is a linear function which can be calculated by existing statistical software packages or programming languages, e.g., R, S, SAS, SPSS, Minitab, Pandas (Python). In terms of the time series analysis method [29], we construct the following actual processing time model of J_j $(j = 1, 2, \ldots, n)$ when it is placed at the r^{th} $(r = 1, 2, \ldots, n_i)$ slot of the i^{th} $(i = 1, 2, \ldots, m)$ group as

$$p_{i,j}^{r} = q_j(1 + b_i \textstyle\sum_{k=1}^{r-1} p_{[k]}^{(i)}) \qquad (2)$$

where $b_i \ge 0$ is the deterioration index of G_i (assume $\sum_{k=1}^{0} p_{[k]}^{(i)} = 0$). Obviously, the makespan of the problem is the sum of the jobs' processing times if $b_i = 0$ for all groups G_i $(i = 1, 2, \ldots, m)$, i.e., the problem can be solved in polynomial time. Therefore, $b_i > 0$ are assumed in the following. Compared to existing theoretical models, Equation (2) is constructed using existing statistical software packages or programming languages. In addition, the assumption that different groups have various deterioration indices is reasonable since it is almost impossible to restore machines to the original state after maintenances.

Let T^M denote the maintenance time and G represents the group scheduling problem. The considered group scheduling problem with both time-dependent deteriorating effects and maintenance can be denoted as $1|p_{i,j}^{r} = q_j(1 + b_i \sum_{k=1}^{r-1} p_{[k]}^{(i)}), T^M, G|C_{\max}$ according to the three-field notation scheme introduced by Graham et al. [30]. Let t_i be the starting time of the first scheduled job in group G_i. It is not hard to obtain the completion time of G_i with $b_i > 0$ for the considered probelm

$$C_i = t_i + \frac{1}{b_i}\Pi_{j=1}^{n_i}(1 + b_i q_i^j) - \frac{1}{b_i} \qquad (3)$$

Equation (3) implies that the makespan calculation can be calculated by Theorem 1.

Theorem 1 *The makespan of the* $1|p_{i,j}^{r} = q_j(1 + b_i \sum_{k=1}^{r-1} p_{[k]}^{(i)}),$ $T^M, G|C_{\max}$ *problem can be calculated as*

$$C_{\max} = \sum_{i=1}^{m} \frac{1}{b_i} \prod_{j=1}^{n_i} (1 + b_i q_i^j) - \sum_{i=1}^{m} \frac{1}{b_i} + \sum_{i=1}^{m-1} s_i \qquad (4)$$

Proof Equation (3) illustrates the completion time of the last job (i.e the $n_i{}^{th}$ job) in the group G_i is $C_i = t_i + \frac{1}{b_i}\Pi_{j=1}^{n_i}(1 + b_i q_i^j) - \frac{1}{b_i}$, i.e.,

$$C_1 = 0 + \frac{1}{b_1}\Pi_{j=1}^{n_1}(1 + b_1 q_1^j) - \frac{1}{b_1}$$

$$C_2 = C_1 + s_1 + \frac{1}{b_2}\Pi_{j=1}^{n_2}(1 + b_2 q_2^j) - \frac{1}{b_2}$$

$$\ldots$$

$$C_m = C_{m-1} + s_{m-1} + \frac{1}{b_m}\Pi_{j=1}^{n_m}(1 + b_m q_m^j) - \frac{1}{b_m}$$

Then $\sum_{i=1}^{m} C_i = \sum_{i=1}^{m-1} C_i + \sum_{i=1}^{m-1} s_i + \sum_{i=1}^{m} \frac{1}{b_i} \prod_{j=1}^{n_i}(1 + b_i q_i^j) - \sum_{i=1}^{m} \frac{1}{b_i}$, i.e., $C_{\max} = C_m = \sum_{i=1}^{m} \frac{1}{b_i} \prod_{j=1}^{n_i}(1 + b_i q_i^j) - \sum_{i=1}^{m} \frac{1}{b_i} + \sum_{i=1}^{m-1} s_i$. $\square$

For simplicity, we denote $\xi_i = \frac{1}{b_i} \prod_{j=1}^{n_i}(1 + b_i q_i^j)$, $\psi(m) = \sum_{i=1}^{m} \xi_i = \sum_{i=1}^{m} \frac{1}{b_i} \prod_{j=1}^{n_i}(1 + b_i q_i^j)$, $s_0 = 0$. Theorem 1 illustrates that the makespan of a schedule depends on both the number of groups and the specific jobs in each group. Then we can translate the proposed problem into the following model.

$$\min \sum_{i=1}^{m} \frac{1}{b_i} \prod_{j=1}^{n} (1 + b_i q_j x_{i,j}) + \sum_{i=1}^{m} (s_{i-1} - \frac{1}{b_i}) \qquad (5)$$

Subject to:

$$\sum_{i=1}^{m} x_{i,j} = 1$$

$$\sum_{j=1}^{n} x_{i,j} \geq 1$$

$$x_{i,j} \in \{0,1\}, i = 1,2,\ldots,m, j = 1,2,\ldots,n$$

where the binary decision variable $x_{i,j} = 1$ means that job j is assigned to group i.

Specifically, $\sum_{i=1}^{m}(s_{i-1} - \frac{1}{b_i})$ is a constant when the number of groups is fixed [7]–[11]. In this case, minimizing $C_{\max}$ is equal to minimizing $\psi(m)$, which implies that the completion time of a group depends on the jobs' normal processing times scheduled in a group and does not depend on their relative positions in the group. In terms of Equation (3), the time complexity of calculating the processing time of the i^{th} group decreases from $O(n_i^2)$ to $O(n_i)$. The worst time complexity of the total completion time decreases from $O(n^2 + m)$ to $O(n + m)$ by Theorem 1.

The problem represented by Equation (5) with a given number of groups is an assignment 0-1 model which is NP-hard [31]. However, the number of jobs in each group in our considered problem can be any number in $[1, n]$ which is more general than the problem with a given number of groups. Therefore, the problem considered in this paper is NP-hard. The proposed problem is non-linear and very complex which means that it is an NP-hard problem. We will solve it with exact, heuristic and meta-heuristic algorithms.

IV. Proposed Heuristics

By Theorem 1, the considered problem $1|p_{i,j}^{r} = q_j(1 + b_i\sum_{k=1}^{r-1} p_{[k]}^{(i)}), T^M, G|C_{\max}$ is equivalently translated into the following problem: A set of jobs $\pi = (J_1, J_2, \cdots, J_n)$ has to be allocated m $(1 \leq m \leq n)$ groups to minimize $C_{\max} = \psi(m) + \sum_{i=1}^{m}(s_{i-1} - \frac{1}{b_i})$. The normal processing time of job J_j is q_i^j in group G_i. b_i and n_i are the deterioration index and the number of jobs in group G_i respectively. $C_{\max}$ depends on the deterioration index b_i, normal processing times q_i^j, maintenance times s_i and the number of groups m. Theorem 1 implies that the makespan of a schedule depends on both m and the specific jobs assigned to each group. The number of groups m is closely related to the number of maintenance, which has an influence on the deteriorating effect. A greater m implies more maintenance or longer maintenance times while it also means less deteriorating effect and shorter processing times. Therefore, it is important to determine an appropriate m to get a balance between maintenance times and deteriorated processing times. On the other hand, $\sum_{i=1}^{m}(s_{i-1} - \frac{1}{b_i})$ is a constant for a given m.

It is natural to obtain a global optimum schedule by searching for optimal values $\psi(m)$ for all m $(m = 1, \ldots, n)$, which is NP-hard. However, we can obtain the near-optimal value of $\psi(m)$ based on the following theorem, which is derived from Hardy's Lemma [32].

Theorem 2 *For two sets $\{x_1, x_2, \cdots, x_n\}$ and $\{y_1, y_2, \cdots, y_n\}$, $\sum_{i=1}^{n} x_i \times y_i$ is minimum if elements of the former set are ordered non-decreasingly by the value of x_i and those of the latter set are sequenced non-increasingly by the value of y_i.*

Proof Assume $x_1 \leq x_2 \leq \cdots \leq x_n$ and $y_1 \geq y_2 \geq \cdots \geq y_n$. Let $z_1 = x_1 \times y_1 + \cdots + x_i \times y_i + x_{i+1} \times y_{i+1} + \cdots + x_n \times y_n$ and $z_2 = x_1 \times y_1 + \cdots + x_i \times y_{i+1} + x_{i+1} \times y_i + \cdots + x_n \times y_n$. Then $z_1 - z_2 = (x_i - x_{i+1}) \times (y_i - y_{i+1}) \leq 0$, i.e., z_1 is minimum. $\square$

However, ξ_i $(i = 1, \ldots, m)$ is the product of $n_i + 1 \geq 2$ (because there is at least one item in each group) items in $\psi(m)$. Although we can calculate the products iteratively (e.g., products of the first two items in each group are sorted in non-decreasing order and are multiplied by the third items according to Theorem 2), it is very difficult to divide the n items into m groups to get the optimum $\psi(m)$ because of the NP-hardness of the problem. In terms of Theorem 2, we propose two batch-based heuristics for the considered problem in this paper: NBA (Near-balanced Batch Allocation) and UBA (Unbalanced Batch Allocation).

All jobs of the set $\mathbb{J} = \{J_1, J_2, \cdots, J_n\}$ are sorted into a sequence $\pi = (J_{[1]}, J_{[2]}, \cdots, J_{[n]})$ by non-increasing order of their normal processing times. The n jobs are partitioned into $\lceil \frac{n}{m} \rceil$ batches in the following way: The first m jobs in π conform the first batch, i.e., $B_1 = (J_{[1]}, \ldots, J_{[m]})$. The last m jobs in π form the second batch, i.e., $B_2 = (J_{[n-m+1]}, \ldots, J_{[n]})$. The second m jobs after B_1 in π constitute the third batch, i.e., $B_3 = (J_{[m+1]}, \ldots, J_{[2m]})$. The second last m jobs before B_2 in π form the fourth batch, i.e., $B_4 = (J_{[n-2m+1]}, \ldots, J_{[n-m]})$. The other jobs in π are divided in this way and the remaining $n - m \times \lfloor \frac{n}{m} \rfloor$ jobs are included in the last batch. For simplicity, we append $\lceil \frac{n}{m} \rceil \times m - n$ dummy jobs with normal processing times 0 to the end of batch $B_{\lceil \frac{n}{m} \rceil}$ so that this batch contains just m jobs with no influence on the final $\psi(m)$.

A. Near-Balanced Batch Allocation

The NBA (Near-balanced Batch Allocation) method assigns every job in a batch to exactly one group, i.e., the difference in the number of jobs inside any two groups is no more than 1. All possible m $(m = 1, \ldots, n)$ are tested in the following way: Let $\xi_i^{(1)} = \frac{1}{b_{[i]}}$ $(i = 1, \ldots, m)$ and $\overrightarrow{\mathbb{G}}^{(k)} = (G_{[1]}^{(k)}, G_{[2]}^{(k)}, \ldots, G_{[m]}^{(k)})$. Initially, maintenance coefficients are sorted in non-increasing order and we get a non-decreasing sequence $(\frac{1}{b_{[1]}}, \frac{1}{b_{[2]}}, \ldots, \frac{1}{b_{[m]}})$ and the corresponding groups form a sorted sequence $\overrightarrow{\mathbb{G}}^{(1)}$. The m jobs in B_1 are sequentially assigned to groups of $\overrightarrow{\mathbb{G}}^{(1)}$. According to Theorem 2, the current $\psi(m) = \sum_{i=1}^{m} \xi_i^{(2)} = \sum_{i=1}^{m} \xi_i^{(1)}(1 + b_{[i]}q_{[i]}^1)$ is the minimum. All m groups are sorted in non-decreasing order of $\xi_i^{(2)}$ $(i = 1, \ldots, m)$ and we obtain the group sequence $\overrightarrow{\mathbb{G}}^{(2)}$. In terms of Theorem 2, the m jobs in B_2 are sequentially assigned to groups of $\overrightarrow{\mathbb{G}}^{(2)}$. The current $\psi(m)$ is calculated by $\psi(m) = \sum_{i=1}^{m} \xi_i^{(3)} = \sum_{i=1}^{m} \xi_i^{(2)}(1 + b_{[i]}q_{[i]}^2)$.

Similarly, all m groups are sorted in non-decreasing order of $\xi_i^{(k)}$ ($i = 1, \ldots, m$; $k = 2, \ldots, \lceil \frac{n}{m} \rceil$) before assigning the jobs of B_k and we obtain the group sequence $\overrightarrow{\mathbb{G}}^{(k)}$. The m jobs in B_k are sequentially assigned to groups of $\overrightarrow{\mathbb{G}}^{(k)}$ and the current $\psi(m)$ is calculated by $\psi(m) = \sum_{i=1}^{m} \xi_i^{(k+1)} = \sum_{i=1}^{m} \xi_i^{(k)}(1 + b_{[i]} q_{[i]}^{k})$.

The value $\min_{m=1,\ldots,n} \psi(m)$ is returned as the final makespan $C_{\max}$. NBA is formally described in Algorithm 1.

Algorithm 1: NBA

```
   /* Near-balanced Batch Allocation    */
1  begin
2      $C_{\max} \leftarrow \infty$;
3      Sort jobs $\{J_1, J_2, \cdots, J_n\}$ by non-increasing order of
         their normal processing times;
4      for $m = 1$ to $n$ do
5          $x \leftarrow 0$;
6          Partition the sorted $n$ jobs into batches $B_1$, $B_2$,
             ..., $B_{\lceil \frac{n}{m} \rceil}$;
7          Sort groups by deterioration indexes $b_1$,$b_2$,...,$b_m$
             in non-increasing order and obtain sequence
             $\overrightarrow{\mathbb{G}}^{(1)}$;
8          for $i = 1$ to $m$ do
9              $\xi_i^{(1)} \leftarrow \frac{1}{b_{[i]}}$;
10         for $k = 1$ to $\lceil \frac{n}{m} \rceil$ do
11             Sequentially assign the $m$ jobs in $B_k$ to the
                 groups of $\overrightarrow{\mathbb{G}}^{(k)}$;
12             for $i = 1$ to $m$ do
13                 $\xi_i^{(k+1)} \leftarrow \xi_i^{(k)}(1 + b_{[i]} q_{[i]}^{k})$;
14             Obtain $\overrightarrow{\mathbb{G}}^{(k+1)}$ by sorting $\xi_1^{(k+1)}$, $\xi_2^{(k+1)}$, ...,
                 $\xi_m^{(k+1)}$ in non-decreasing order;
15         for $i = 1$ to $m$ do
16             $x \leftarrow x + \xi_i^{\lceil \frac{n}{m} \rceil} + s_{i-1} - \frac{1}{b_i}$;
17         if $x < C_{\max}$ then
18             $C_{\max} \leftarrow x$;
19     return $C_{\max}$.
```

The time complexity of step 3 is $O(n \log n)$, that of steps 7 and 14 is $O(m \log m)$ and that of step 14 is $O(n \log m)$. Therefore the time complexity of NBA is $O(n^2 \log m)$.

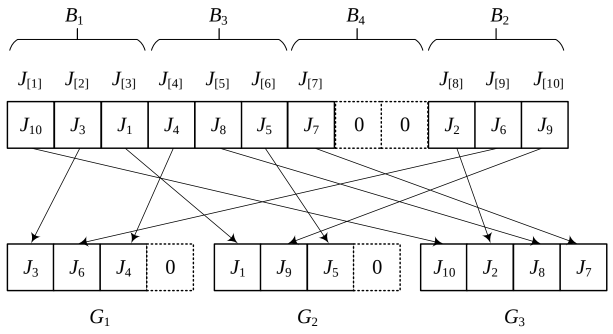
To illustrate the above procedure, the following example is considered for $m = 3$: $n = 10$, $q_1 = 72$, $q_2 = 40$, $q_3 = 76$, $q_4 = 71$, $q_5 = 42$, $q_6 = 7$, $q_7 = 41$, $q_8 = 61$, $q_9 = 6$, $q_{10} = 87$, $b_1 = 0.0005$, $b_2 = 0.00049$, $b_3 = 0.00051$, $s_0 = 0$, $s_1 = 12$, $s_2 = 40$.

According to the NBA heuristic, the 10 jobs are sorted by their processing times in non-increasing order and we obtain the sequence $(J_{[1]}, J_{[2]}, \cdots, J_{[10]}) = (J_{10}, J_3, J_1, J_4, J_8, J_5, J_7, J_2, J_6, J_9)$. Using the above batch partitioning method, the jobs are partitioned into four batches: $B_1 = (J_{10}, J_3, J_1)$, $B_2 = (J_2, J_6, J_9)$, $B_3 = (J_4, J_8, J_5)$, $B_4 = (J_7)$. Two jobs with processing times 0 are appended to B_4 i.e., B_4 becomes $(J_7, 0, 0)$. $\overrightarrow{\mathbb{G}}^{(1)} = (G_3, G_1, G_2)$ is obtained by sorting deterioration indexes b_1,b_2,b_3 in non-increasing order. The procedure of the jobs allocated to the groups is illustrated in Table II. The final schedule is $G_1 = (J_3, J_6, J_4)$, $G_2 = (J_1, J_9, J_5)$, $G_3 = (J_{10}, J_2, J_8, J_7)$ with the makespan being 569.877.

TABLE II: Allocating jobs to groups in the example of NBA heuristic

Batch	Group	Job allocation	ξ values
$B_1 = (J_{10}, J_3, J_1)$	$\overrightarrow{\mathbb{G}}^{(1)} = (G_3, G_1, G_2)$	$G_1 = (J_3)$	2076
		$G_2 = (J_1)$	2112.816
		$G_3 = (J_{10})$	2047.784
$B_2 = (J_2, J_6, J_9)$	$\overrightarrow{\mathbb{G}}^{(2)} = (G_3, G_1, G_2)$	$G_1 = (J_3, J_6)$	2083.266
		$G_2 = (J_1, J_9)$	2119.028
		$G_3 = (J_{10}, J_2)$	2089.559
$B_3 = (J_4, J_8, J_5)$	$\overrightarrow{\mathbb{G}}^{(3)} = (G_1, G_3, G_2)$	$G_1 = (J_3, J_6, J_4)$	2157.222
		$G_2 = (J_1, J_9, J_5)$	2162.638
		$G_3 = (J_{10}, J_2, J_8)$	2154.565
$B_4 = (J_7, 0, 0)$	$\overrightarrow{\mathbb{G}}^{(4)} = (G_3, G_1, G_2)$	$G_1 = (J_3, J_6, J_4, 0)$	2157.222
		$G_2 = (J_1, J_9, J_5, 0)$	2162.638
		$G_3 = (J_{10}, J_2, J_8, J_7)$	2199.617

The schematic graph of the NBA procedure for the example is illustrated in Figure 1.



B1
B3
B4
B2
J[1] J[2] J[3] J[4] J[5] J[6] J[7] J[8] J[9] J[10]
J10 J3 J1 J4 J8 J5 J7 0 0 J2 J6 J9
J3 J6 J4 0
J1 J9 J5 0
J10 J2 J8 J7
G1
G2
G3

Fig. 1: The schematic graph of NBA.

B. Unbalanced Batch Allocation

Unlike NBA, UBA (Unbalanced Batch Allocation) appends each job in a batch to the group with the least sum of actual processing times. All possible m ($m = 1, \ldots, n$) are tested in the following way: Initially, deterioration indexes are sorted in non-increasing order and we get a non-decreasing sequence $(\frac{1}{b_{[1]}}, \frac{1}{b_{[2]}}, \ldots, \frac{1}{b_{[m]}})$ and the corresponding sorted groups $\overrightarrow{\mathbb{G}} = (G_{[1]}, G_{[2]}, \ldots, G_{[m]})$. Let $\xi_i = \frac{1}{b_{[i]}}$ and $n_i = 1$ ($i = 1, \ldots, m$). The m jobs in B_1 are sequentially assigned to groups of $\overrightarrow{\mathbb{G}}$. According to Theorem 2, the current $\psi(m) = \sum_{i=1}^{m} \xi_i(1 + b_{[i]} q_{[i]}^{1})$ is the minimum. For each job of batch B_k ($k = 2, \ldots, \lceil \frac{n}{m} \rceil$), we append it to $G_{[j]}$ where

$j = \arg\min_{i=1,\ldots,m}\{\xi_i\}$. $n_{[j]} = n_{[j]} + 1$ and ξ_j is updated by $\xi_j(1+b_{[j]}q_{[j]}^{n_{[j]}})$. After all jobs are allocated, $\psi(m)$ is calculated by $\psi(m) = \sum_{i=1}^{m}\xi_i$. The value $\min_{m=1,\ldots,n}(\psi(m) + s_{i-1} - \frac{1}{b_i})$ is returned as the final makespan $C_{\max}$. UBA is formally described in Algorithm 2.

Algorithm 2: UBA

```
   /* Unbalanced Batch Allocation */
1  begin
2      $C_{\max} \leftarrow \infty$;
3      Sort jobs $\{J_1, J_2, \cdots, J_n\}$ by non-increasing order of their normal processing times;
4      for $m = 1$ to $n$ do
5          $x \leftarrow 0$;
6          Partition the sorted $n$ jobs into batches $B_1$, $B_2$, ..., $B_{\lceil \frac{n}{m} \rceil}$;
7          Sort groups by deterioration indexes $b_1$,$b_2$,...,$b_m$ in non-increasing order and obtain sequence $\overrightarrow{\mathbb{G}}$;
8          for $i = 1$ to $m$ do
9              $\xi_i \leftarrow \frac{1}{b_{[i]}}$; $n_i \leftarrow 1$;
10         Sequentially assign the $m$ jobs of $B_1$ to the groups of $\overrightarrow{\mathbb{G}}$;
11         $\xi^* \leftarrow \infty$;
12         for $i = 1$ to $m$ do
13             $\xi_i \leftarrow \xi_i(1 + b_{[i]}q_{[i]}^1)$;
14             if $\xi^* > \xi_i$ then
15                 $\xi^* \leftarrow \xi_i$; $j \leftarrow i$;
16         for $k = 2$ to $\lceil \frac{n}{m} \rceil$ do
17             for $i = 1$ to $m$ do
18                 Append the current job of $B_k$ to $G_{[j]}$;
19                 $n_{[j]} \leftarrow n_{[j]} + 1$;
20                 $\xi_j \leftarrow \xi_j(1 + b_{[j]}q_{[j]}^{n_{[j]}})$;
21                 $\xi^* \leftarrow \infty$;
22                 for $l = 1$ to $m$ do
23                     if $\xi^* > \xi_l$ then
24                         $\xi^* \leftarrow \xi_l$; $j \leftarrow l$;
25         for $i = 1$ to $m$ do
26             $x \leftarrow x + \xi_i + s_{i-1} - \frac{1}{b_i}$;
27         if $C_{\max} > x$ then
28             $C_{\max} \leftarrow x$;
29     return $C_{\max}$.
```

The time complexity of step 3 is $O(n \log n)$, that of step 7 is $O(m \log m)$, and that of steps 9, 15 and 26 is $O(m)$. The time complexity of step 24 is $O(mn)$. Since $n \geq m$, the time complexity of UBA is $O(mn^2)$.

To illustrate the previous procedure, the above example is used. According to the UBA heuristic, the 10 jobs are sorted by their processing times in non-increasing order and again we obtain the sequence $(J_{[1]}, J_{[2]}, \cdots, J_{[10]}) = (J_{10}, J_3, J_1, J_4, J_8, J_5, J_7, J_2, J_6, J_9)$. Using the above batch partitioning method, the jobs are segmented into four batches: $B_1 = (J_{10}, J_3, J_1)$, $B_2 = (J_2, J_6, J_9)$, $B_3 = (J_4, J_8, J_5)$, $B_4 = (J_7, 0, 0)$. $\overrightarrow{\mathbb{G}} = (G_3, G_1, G_2)$ is obtained by sorting deterioration indexes b_1,b_2,b_3 in non-increasing order. J_{10}, J_3, J_1 are allocated to G_3,G_1,G_2 respectively and the following procedure of the jobs allocated to the groups is illustrated in Table III. The final schedule is $G_1 = (J_3, J_6, J_9, J_4)$, $G_2 = (J_1, J_5)$, $G_3 = (J_{10}, J_2, J_8, J_7)$ with the makespan being 570.009.

TABLE III: Allocating jobs to groups in the example of UBA heuristic

Batch	Current Job	Job allocation	ξ values
$B_1 = (J_{10}, J_3, J_1)$		$G_1 = (J_3)$	2076
		$G_2 = (J_1)$	2112.816
		$G_3 = (J_{10})$	**2047.784**
$B_2 = (J_2, J_6, J_9)$	J_2	$G_1 = (J_3)$	**2076**
		$G_2 = (J_1)$	2112.816
		$G_3 = (J_{10}, J_2)$	2089.559
	J_6	$G_1 = (J_3, J_6)$	**2083.266**
		$G_2 = (J_1)$	2112.816
		$G_3 = (J_{10}, J_2)$	2089.559
	J_9	$G_1 = (J_3, J_6, J_9)$	**2089.516**
		$G_2 = (J_1)$	2112.816
		$G_3 = (J_{10}, J_2)$	2089.559
$B_3 = (J_4, J_8, J_5)$	J_4	$G_1 = (J_3, J_6, J_9, J_4)$	2163.694
		$G_2 = (J_1)$	2112.816
		$G_3 = (J_{10}, J_2)$	**2089.559**
	J_8	$G_1 = (J_3, J_6, J_9, J_4)$	2163.694
		$G_2 = (J_1)$	**2112.816**
		$G_3 = (J_{10}, J_2, J_8)$	2154.565
	J_5	$G_1 = (J_3, J_6, J_9, J_4)$	2163.694
		$G_2 = (J_1, J_5)$	2156.298
		$G_3 = (J_{10}, J_2, J_8)$	**2154.565**
$B_4 = (J_7, 0, 0)$	J_7	$G_1 = (J_3, J_6, J_9, J_4)$	2163.694
		$G_2 = (J_1, J_5)$	2156.298
		$G_3 = (J_{10}, J_2, J_8, J_7)$	2199.617

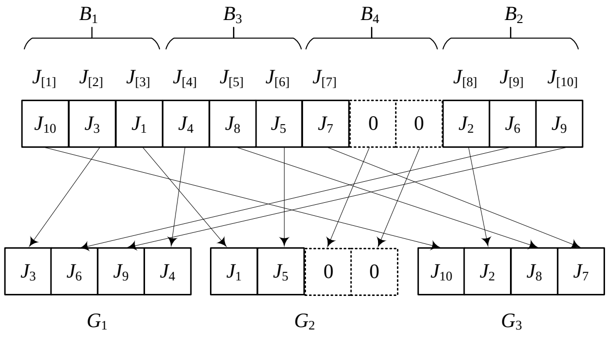
The schematic graph of the UBA procedure for the example is illustrated by Figure 2.

C. Intuitive Enumeration Method

To show how close the solutions obtained by the proposed NBA and UBA are to the optimum solution of the problem, in addition, the proposed problem is high non-linearity, we develop an intuitive enumeration method. Note that the enumeration method is time-consuming when the problem size increases in terms of the following theorem.

Theorem 3 *The number of possible solutions to the problem* $1|p_{i,j}^r = q_j(1 + b_i \sum_{k=1}^{r-1} p_{[k]}^{(i)}), T^M, G|C_{\max}$ *is* $(n!)2^{n-1}$.

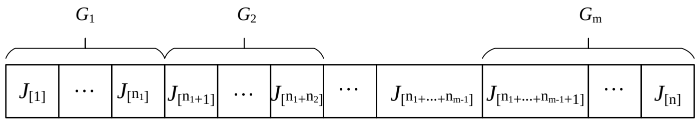
Proof There are $n!$ permutations for the job set $\mathbb{J} = \{J_1, J_2, \cdots, J_n\}$. Each permutation contains n jobs and could



B_1
B_3
B_4
B_2
$J_{[1]}$ $J_{[2]}$ $J_{[3]}$ $J_{[4]}$ $J_{[5]}$ $J_{[6]}$ $J_{[7]}$ $J_{[8]}$ $J_{[9]}$ $J_{[10]}$
J_{10} J_3 J_1 J_4 J_8 J_5 J_7 0 0 J_2 J_6 J_9
J_3 J_6 J_9 J_4 J_1 J_5 0 0 J_{10} J_2 J_8 J_7
G_1
G_2
G_3

Fig. 2: The schematic graph of UBA.

be represented as $\pi = (J_{[1]}, J_{[2]}, \cdots, J_{[n]})$. We denote the set of job permutations as Ω. For a given $\pi \in \Omega$, the n jobs are segmented into m groups ($m = 1, \ldots, n$). Let the number of jobs in group i ($1 \le i \le m$) be n_i ($n_i > 0$). The m groups of $\pi = (J_{[1]}, J_{[2]}, \cdots, J_{[n]})$ are denoted as $(J_{[1]}, J_{[2]}, \cdots, J_{[n_1]})$, $(J_{[n_1+1]}, J_{[n_1+2]}, \cdots, J_{[n_1+n_2]})$, $\cdots$, $(J_{[n_{m-1}+1]}, J_{[n_{m-1}+2]}, \cdots, J_{[n_{m-1}+n_m]})$, as shown in Figure 3. We can regard the n jobs and m groups as being separated by $n-1$ and $m-1$ slots respectively. For each m, the segmentation can be viewed as selecting $m-1$ slots from $n-1$ slots and there are C_{n-1}^{m-1} candidates. For all $m = 1, \ldots, n$, there are $\sum_{m=1}^{n} C_{n-1}^{m-1} = 2^{n-1}$ candidates. Therefore, there are $(n!)2^{n-1}$ possible solutions for the $1|p_{i,j}^r = q_j(1 + b_i \sum_{k=1}^{r-1} p_{[k]}^{(i)}), T^M, G|C_{\max}$ problem. $\square$



G_1
G_2
G_m
$J_{[1]}$ $\cdots$ $J_{[n_1]}$ $J_{[n_1+1]}$ $\cdots$ $J_{[n_1+n_2]}$ $\cdots$ $J_{[n_1+\ldots+n_{m-1}]}$ $J_{[n_1+\ldots+n_{m-1}+1]}$ $\cdots$ $J_{[n]}$

Fig. 3: The jobs scheduled in the groups

Obviously, the most intuitive method for the $1|p_{i,j}^r = q_j(1+ b_i \sum_{k=1}^{r-1} p_{[k]}^{(i)}), T^M, G|C_{\max}$ problem is enumeration, which is called IEM (Intuitive Enumeration Method) in this paper and formally described in Algorithm 3. In fact, IEM is only suitable for small size instances according to Theorem 3 and it is time-consuming when $n > 10$. Therefore, we only compare the proposed NBA and UBA with IEM on small size instances.

D. Evaluation of the Proposed Heuristics

To the best of our knowledge, there is no scheduling problem studied so far in the literature with the same features as the problem considered in this paper. The three proposed algorithms (NBA, UBA and IEM) are implemented in Java and run on an Intel(R) Core(TM) i7-4770 CPU @3.40GHz computer with 8GB RAM on Windows Server 2008 R2 standard.

Being a new problem, there are no existing benchmarks. To fairly compare different procedures, instances are randomly generated according to the characteristics of the problem. Considering the CPU time requirements of IEM, we compare the three algorithms on small size instances with $n \in \{3, 4, 5, 6, 7, 8, 9, 10\}$ in this section (comparisons on large size instances will be conducted later). We generalize the case given in [33] with fixed $b_i = 0.01$ ($i = 1, \ldots, m$). In this paper, we consider the cases where b_i takes a value randomly from [0, U] for each U in $\{0.005, 0.006, 0.007, 0.008, 0.009, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06\}$. The processing time of each job is a randomly generated integer in [1,99] with a uniform distribution. Maintenance times are random integers respectively generated in [1, Q] with a uniform distribution where $Q \in \{99, 199\}$, i.e., maintenance times are randomly generated from [1,99] and [1,199] respectively. For each combination of n, U and the maintenance time type Q are randomly generated. In addition, considering the proposed NBA, UBA and IEM are exact, we do not replicate the experiment under the same combination. Therefore, there are $8 \times 11 \times 2 = 176$ instances in total.

The RPD (Relative Percentage Deviation) is adopted to measure the effectiveness of the proposed NBA, UBA and IEM. Let $C_{\max}(H)$ be the makespan obtained by heuristic H and C^* be the optimum makespan for an instance. RPD is

Algorithm 3: IEM

```
begin
    $C_{\max} \leftarrow \infty$, $A \leftarrow \emptyset$;
    $\pi \leftarrow (J_1, J_2, \cdots, J_n)$;
    Construct the job permutation set $A$ of $\pi$;
    for $m = 1$ to $n$ do
        $\vec{a} \leftarrow (a_{[1]}, a_{[2]}, \cdots, a_{[n-1]}) = (0, 0, \cdots, 0)$;
        for $i = 1$ to $m$ do
            $a_{[i]} \leftarrow 1$;
        $B \leftarrow \{\vec{a}\}$;
        Construct the combination set $B$ of $C_{n-1}^{m-1}$;
        foreach $(J_{[1]}, J_{[2]}, \cdots, J_{[n]}) \in A$ do
            foreach $(a_{[1]}, a_{[2]}, \cdots, a_{[n-1]}) \in B$ do
                $\psi(m) \leftarrow 0$ ;
                for $i = 1$ to $m$ do
                    $G_i \leftarrow \varnothing$, $\xi_i \leftarrow \frac{1}{b_i}$;
                $j \leftarrow 1$;
                for $i = 1$ to $n-1$ do
                    $G_j \leftarrow G_j \bigcup \{J_{[i]}\}$, $\xi_j \leftarrow \xi_j(1 + b_j q_{[i]})$;
                    if $a_{[i]} = 1$ then
                        $j \leftarrow j + 1$;
                $G_m \leftarrow G_m \bigcup \{J_{[n]}\}$;
                $\xi_m \leftarrow \xi_m(1 + b_m q_{[n]})$;
                for $i = 1$ to $m$ do
                    $\psi(m) \leftarrow \psi(m) + \xi_i + s_{i-1} - \frac{1}{b_i}$;
                if $C_{\max} > \psi(m)$ then
                    $C_{\max} \leftarrow \psi(m)$;
    return $C_{\max}$.
```

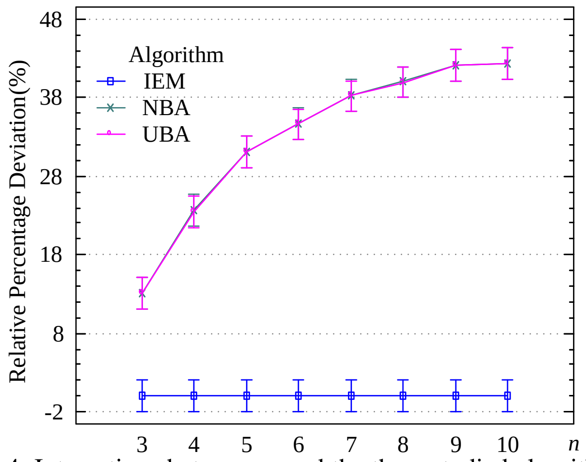


Fig. 4: Interactions between n and the three studied algorithms at 95% confidence Tukey HSD intervals for RPD.

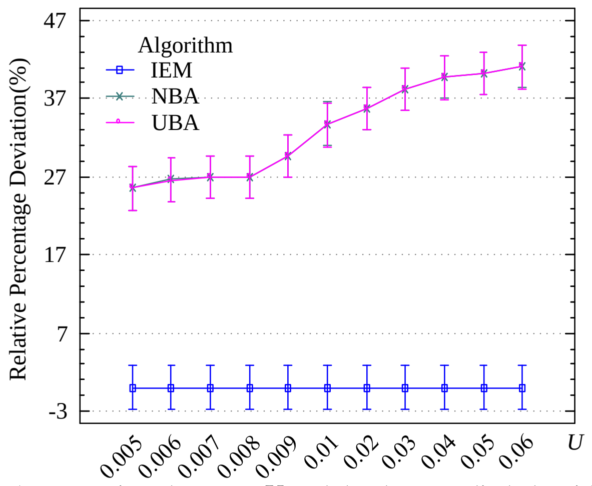


Fig. 5: Interactions between U and the three studied algorithms at 95% confidence Tukey HSD intervals for RPD.

defined as follows:

$$RPD(H) = \frac{C_{\max}(H) - C^*}{C_{\max}(H)} \times 100\%$$

First, we show the influence of parameters n and U on NBA and UBA. To test for statistical significance, a multi-factor analysis of variance (ANOVA) method is carried out. The response variable is the RPD. The three main hypotheses (normality, homoscedasticity, and independence of the residuals) are checked from the residuals of the experiments. All three hypotheses are acceptable from this analysis. Since all the p-values in the experiments are close to zero, they are not given in this paper.

Interactions between n and the three studied algorithms at the 95% confidence Tukey HSD intervals for RPD are depicted in Figure 4. From Figure 4, we can see that RPDs of both NBA and UBA increase with the number of jobs n with similar tendencies. n has little influence on IEM because IEM is an enumerative method and optimum solutions are obtained in all cases. The differences are not significant when $3 \leq n \leq 10$.

Interactions between U and the three studied algorithms at 95% confidence Tukey HSD intervals for RPD are depicted in Figure 5. Figure 5 demonstrates that U has little influence on the effectiveness of NBA and UBA with similar tendencies. RPDs of both NBA and UBA increase with the increase in U. However, the increasing differences are not significant except in the $U \in \{0.008,\ 0.05\}$ cases.

Interactions between Q and the three studied algorithms at 95% confidence Tukey HSD intervals for RPD are depicted in Figure 6. Figure 6 illustrates that Q has a great influence on the effectiveness of NBA and UBA with similar tendencies. The RPDs of both NBA and UBA decrease dramatically when Q increases from 99 to 199.

NBA, UBA, IEM are compared to PTA [25], NEW_FF and NEW_BF [21] on average RPD (ARPD) and CPU time (in seconds) in Table IV. ARPDs of NBA and UBA increase with an increase in n which is in accordance with the results indicated in Figure 4. On average, ARPDs of NBA and UBA

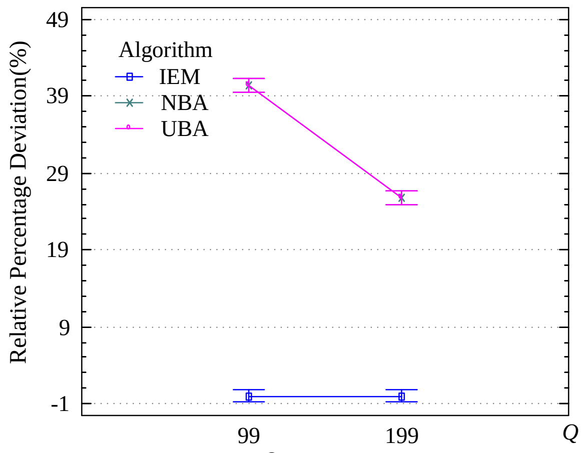


Fig. 6: Interactions between Q and the three studied algorithms at 95% confidence Tukey HSD intervals for RPD.

are 33.15% and 33.11% respectively. Although the differences are big, this is not always the case with the absolute gaps. For example, makespans of NBA, UBA, IEM, PTA, NEW_FF and NEW_BF are 569.88, 570.01, 540.61, 568.17, 745 and 745 respectively for an instance with $U = 0.008$, $Q = 199$ and $n = 6$ which are depicted in Figure 7. The gaps are merely about 21. However, IEM is the most time-consuming algorithm with the CPU time increasing exponentially depending on the number of jobs n. 667.2s of CPU time is needed by IEM when $n = 10$ while CPU times of the other compared algorithms are negligible for all the $n \leq 10$ cases. Although it seems that NBA, UBA and PTA perform similarly for small instances, it will be shown later than for the proposed IG meta-heuristic, NBA is a superior heuristic when used for initialization. NEW_FF and NEW_BF are outperformed by the other four algorithms because they are heuristics for periodical cases.

TABLE IV: Performance comparisons on NBA, UBA, IEM, PTA, NEW_FF and NEW_BF.

n	NBA		UBA		IEM		PTA		NEW_FF		NEW_BF	
	ARPD(%)	CPU Time(s)	ARPD(%)	CPU Time(s)	ARPD(%)	CPU Time(s)	ARPD(%)	CPU Time(s)	ARPD(%)	CPU Time(s)	ARPD(%)	CPU Time(s)
3	13.13	< 1	13.13	< 1	0	< 1	13.18	< 1	30.64	< 1	30.64	< 1
4	23.63	< 1	23.48	< 1	0	< 1	23.48	< 1	38.88	< 1	38.92	< 1
5	31.05	< 1	31.04	< 1	0	< 1	30.81	< 1	46.17	< 1	46.28	< 1
6	34.71	< 1	34.63	< 1	0	< 1	34.64	< 1	47.26	< 1	47.28	< 1
7	38.24	< 1	38.21	< 1	0	< 1	38.10	< 1	50.27	< 1	50.34	< 1
8	40.00	< 1	39.98	< 1	0	1.71	39.87	< 1	51.80	< 1	51.87	< 1
9	42.08	< 1	42.07	< 1	0	36.23	41.95	< 1	53.00	< 1	53.05	< 1
10	42.35	< 1	42.31	< 1	0	667.22	42.20	< 1	53.09	< 1	53.19	< 1
Average	33.15		33.11		0		33.03		46.39		46.45	

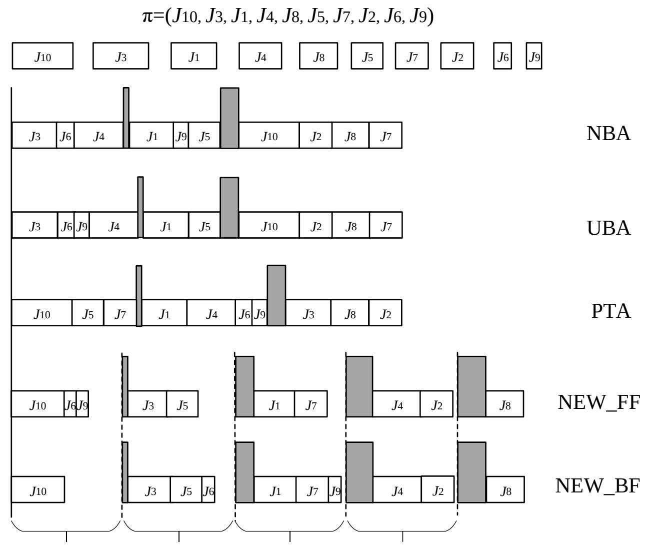


Fig. 7: Gantt Charts of the example using NBA, UBA, IEM, PTA, NEW_FF and NEW_BF.

V. Proposed Iterated Greedy Heuristic

The proposed heuristics NBA and UBA get near-optimal solutions for small size problems. As shown in Figure 4, the effectiveness of the two heuristics decreases quickly with the increase in the number of jobs n. It is desirable to develop effective algorithms for large size problems.

Ruiz and Stützle [34], [35], Pan and Ruiz [36] and Ribas et al. [37] showed that IG (Iterated Greedy) algorithms are effective for flowshop machine scheduling problems with makespan minimization. Rodriguez et al. [38] also demonstrated that IG algorithms are effective for unrelated parallel machine scheduling problems with makespan minimization. Therefore, we present an IG framework based algorithm for the considered machine scheduling problem in this paper. This consists of three basic phases: Initial Sequence Construction, Destruction and Reconstruction and LocalSearch. The framework of the proposed IG is depicted in Algorithm 4. The different operators are explained in the following sections.

Algorithm 4: Iterated Greedy Heuristic

```
1 begin
2     $\pi \leftarrow$ InitialSolution;
3     while (Termination criterion not satisfied) do
4         $\pi' \leftarrow$ DestructionReconstruction($\pi, d$);
5         $\pi'' \leftarrow$ LocalSearch($\pi'$);
6         $\pi \leftarrow$ AcceptanceCriterion($\pi$, $\pi''$);
7     return $\pi$.
```

A. Initial Sequence Construction (ISC)

It is common to use effective heuristics to obtain good initial solutions [37]. To construct an initial solution by allocating n jobs to m groups, the proposed heuristics NBA and UBA are adopted. In addition, a RSC (Random Sequence Construction) method is used to construct random initial solutions for the considered problem in this paper. In RSC, m is a random number between 1 and n and all jobs are randomly allocated to the m groups.

B. Destruction & Reconstruction

To enhance the intensification of the proposed IG algorithm, the Destruction & Reconstruction (DR) process is carried out on sequences. Sequence π is destructed by randomly selecting and removing d different jobs. Two subsequences $\pi^R = (\pi^R_{[1]}, \dots, \pi^R_{[d]})$ and π^D denote the removed d jobs and the remaining $n-d$ jobs respectively. They keep the same order as in π. In this paper, a new Destruction & Reconstruction is proposed for the considered problem, of which the destruction process is identical to that developed by Ruiz and Stützle [34], [35] whereas the reconstruction process is different. When a new job sequence π' is reconstructed, all jobs of π^R are sequentially reinserted back into π^D during the reconstruction process in the following way: m subsequences are generated by trying to reinsert $\pi^R_{[i]}$ $(i = 1, \dots, d)$ into each group of π^D. Makespans of the m subsequences are calculated by Equation (4). The subsequence with the minimum makespan is selected as π'. The reconstruction stops after d iterations or $\pi^R_{[d]}$ is tried. The DR is formally described in Algorithm 5. Though the makespan calculation of Step 8 can be done in $O(1)$ based on the result of the initial π^D, the time complexity of the makespan calculation of π^D is $O(n-d)$. Therefore, the time complexity of the proposed DR is $O(mnd)$.

Algorithm 5: DR(π, d)

```
/* Destruction & Reconstruction */
1  begin
2      Generate subsequence $\pi^R$ by randomly removing $d$ jobs from $\pi$;
3      Denote the remaining subsequence with $n-d$ jobs as $\pi^D$;
4      for $i = 1$ to $d$ do
5          $Temp \leftarrow \infty$, $\varpi^{(0)} \leftarrow \pi^D$;
6          for $k = 1$ to $m$ do
7              Construct $\varpi^{(k)}$ by inserting job $\pi^R_{[i]}$ to group $G_k$ of $\pi^D$;
8              Calculate $C_{\max}(\varpi^{(k)})$ using Equation (4);
9              if $C_{\max}(\varpi^{(k)}) < Temp$ then
10                 $\varpi^{(0)} \leftarrow \varpi^{(k)}$, $Temp \leftarrow C_{\max}(\varpi^{(k)})$;
11         $\pi^D \leftarrow \varpi^{(0)}$;
12     $\pi \leftarrow \pi^D$;
13     return $\pi$.
```

C. LocalSearch

To improve the diversification of the proposed IG algorithm, the LocalSearch procedure is conducted on a sequence. Every job $J_{[j]}$ is selected from π. Construct $\pi^{'}$ by deleting J_j from π. Construct $\pi^{''}$ by inserting J_j into every group from $\pi^{'}$. π will be replaced by $\pi^{''}$ if $C_{\max}(\pi^{''})$ is less than $C_{\max}(\pi)$. The time complexity of LocalSearch is $O(mn)$. LocalSearch is formally described in Algorithm 6.

Algorithm 6: LocalSearch

```
1  begin
2      for $j = 1$ to $n$ do
3          if (the number of the group which $J_j$ is inside do not equal to 1) then
4              Construct $\pi^{'}$ by deleting $J_j$ from $\pi$;
5              for $i = 1$ to $m$ do
6                  Construct $\pi^{''}$ by inserting $J_j$ in $G_i$ from $\pi^{'}$;
7                  Calculate $C_{\max}(\pi^{''})$ using Equation (4);
8                  if $C_{\max}(\pi^{''}) < C_{\max}(\pi)$ then
9                      $\pi \leftarrow \pi^{''}$;
10                 return $\pi$.
```

D. Acceptance Criterion

After performing the above three operators on an initial solution π, a new solution π' is obtained. If π' is better than the incumbent best solution π^*, both π^* and π are replaced by π'. If π' is not better than π^* but better than π, π is replaced by π'. Otherwise π is replaced by π' with a certain probability $e^{-\frac{(C_{\max}(\pi^c)-C_{\max}(\pi))}{Temp}}$, which is similar to the Simulated Annealing acceptance criterion adopted in [34], [35]. The process is repeated until a certain termination criterion is met. The acceptance criterion is presented in Algorithm 7.

Algorithm 7: Acceptance Criterion

```
   Input: The current solution $\pi$, the incumbent best solution $\pi^*$, the newly generated solution $\pi'$.
1  begin
2      if $C_{\max}(\pi') < C_{\max}(\pi)$ then
3          $\pi \leftarrow \pi'$;
4          if $C_{\max}(\pi') < C_{\max}(\pi^*)$ then
5              $\pi^* \leftarrow \pi'$;
6      else
7          Generate a random number $\lambda \in [0,1]$;
8          if $\lambda < e^{-\frac{C_{\max}(\pi')-C_{\max}(\pi)}{Temp}}$ then
9              $\pi \leftarrow \pi'$;
10     return $\pi$, $\pi^*$.
```

VI. PERFORMANCE EVALUATION

Since no comprehensive benchmark of instances is available for the problem under study, we generate random instances according to the same parameters as in the small size case. The instance size n now takes values from $\{10, 50, 100, 150, 200, 300\}$. The proposed algorithm is run in the environment with the same configurations as in the previous sections.

A. Parameter & Component Calibration

Before the algorithm evaluation, we calibrate the three parameters and components to construct initial solutions. Three ISC components (NBA, UBA and RSC) are tested to generate initial solutions. The number of removed jobs $d = \gamma \times n$ in the Destruction & Reconstruction phase takes values from $\gamma \in \{5\%, 10\%, 20\%, 40\%\}$. The $Temp$ in the acceptance criterion $Temp = \eta \times \frac{\sum_{i=1}^{n} q_i}{n}$ takes values $\eta \in \{1, 2, 4\}$. We use the number of consecutive iterations without improvement $\theta \in \{5, 15, 30\}$ as the termination criterion. Therefore, there are $3 \times 4 \times 3 \times 3 = 108$ combinations. For each combination and every size n, we generate 5 random instances different from the final testing instances, Q and U take the same values as in the small size case i.e., $Q \in \{99, 199\}$ and $U \in \{0.005, 0.006, 0.007, 0.008, 0.009, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06\}$. Therefore, $108 \times 6 \times 5 \times 2 \times 11 = 71280$ results are obtained. ARPD defined above is adopted to measure the performance. Experimental results show that Q has a similar influence on the performance of the proposed IG as compared with the small size case. The means plots and 95% confidence

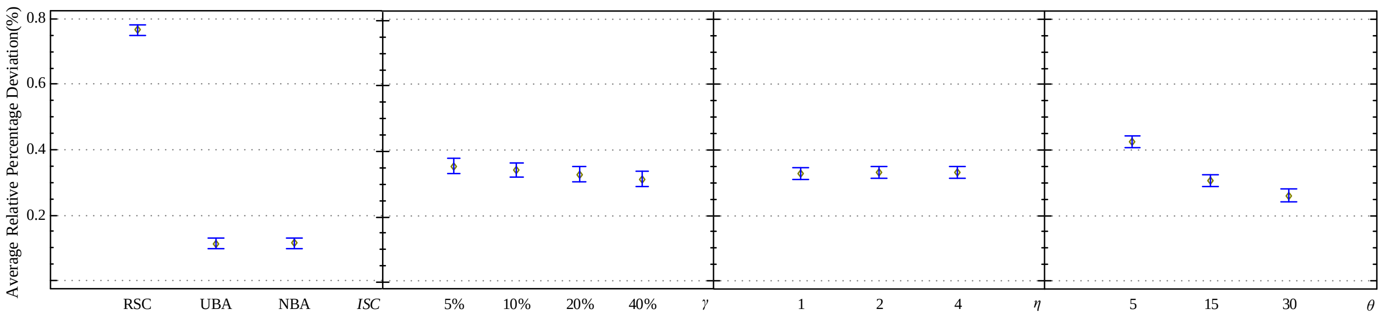


Fig. 8: Means plots and Tukey HSD confidence intervals at the 95% confidence level for ARPDs of ISC, γ, η and θ.

level Tukey HSD (Honest Significant Difference) intervals for ISC, γ, η and θ are depicted in Figure 8.

Figure 8(a) demonstrates that the differences between the three ISC component ARPDs are statistically significant. NBA and UBA can always obtain better solutions than RSC. In addition, ARPDs of the proposed IG with NBA and UBA adopted to construct initial solutions yield superior solutions than using RSC. Also, NBA is statistically similar to UBA.

Figure 8(b) illustrates that ARPD decreases with an increase in γ. Though ARPDs are different for various γ, the differences are not statistically significant. The average ARPD is only 0.31% when $\gamma = 40\%$ which is small enough to be acceptable. Therefore, we take $\gamma = 40\%$ in the following experiments.

Figure 8(c) shows that η does not have a clear influence on the proposed IG and the differences are small. The RPD is about 0.33% when $\eta = 2$, which is adopted in the following experiments.

Figure 8(d) implies that different termination criteria exert a great influence on the performance of the proposed IG. This is expected as with longer CPU times solutions improve. The ARPD differences are statistically significant between 5 and the other two (15 and 30). The average ARPD of the proposal when $\theta = 30$ is much better than that when $\theta = 5$ because a longer computation duration has a higher possibility of finding better solutions.

The means plots and 95% confidence level Tukey HSD intervals for θ is depicted in Figure 9 as an interaction with the ISC algorithm. We can observe from Figure 9 that the differences between the ARPDs of RSC and those of the other two ISC components are statistically significant for θ cases. ARPD of the proposed IG with NBA and UBA adopted to construct initial solutions are clearly better than the algorithm using RSC.

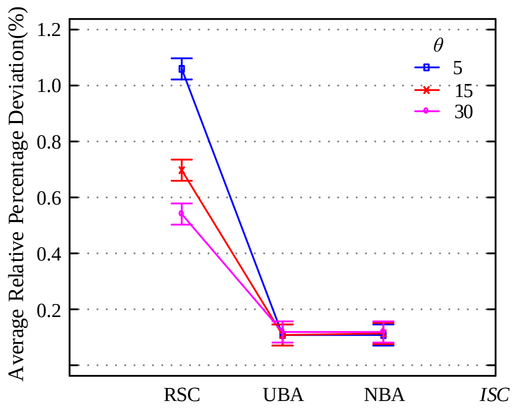


Fig. 9: Interactions between θ and the three studied algorithms at 95% confidence Tukey HSD intervals for ARPD.

B. Performance Evaluation

To further illustrate the performance of the proposed IG_NBA and IG_UBA algorithms, we compare them against PTA [25], NEW_FF and NEW_BF [21] . In the proposed IG algorithms, initial solutions are constructed by NBA and UBA, $\gamma = 40\%$, $\eta = 2$, $\theta = 30$. The compared algorithms are run on instances with all instance parameters n, Q and U as mentioned above. 5 instances are randomly generated for each combination of the involved 6 parameters. The effectiveness (measured by average ARPD) and efficiency (measured by computation time in seconds) of the compared algorithms are shown in Table V.

Table V indicates that APRDs of the proposed IG algorithms are 0.16% on all instance sizes on average, the ARPD of PTA is 0.27%, ARPDs of New_FF and New_BF are 13.23% and 13.26% respectively. It seems the differences are big. Moreover, the two proposed IG algorithms spend much less time than PTA and New_FF with an increase in n but a little more than New_BF. However, IG_NBA, IG_UBA and New_BF spend less than 30s even $n = 300$ which is much acceptable in practical applications.

VII. Conclusions

In this paper, we considered time-dependent deteriorating effects in a type of single machine group scheduling problem which is closer to practical cases than previous studies. According to the time series analysis technique, a linear effect function between the actual processing time and the normal processing time of a job was constructed. We proved that the

TABLE V: Performance comparisons on IG_NBA, IG_UBA, PTA, New_FF and New_BF.

n	IG_NBA		IG_UBA		PTA		NEW_FF		NEW_BF	
	ARPD(%)	CPU Time(s)	ARPD(%)	CPU Time(s)	ARPD(%)	CPU Time(s)	ARPD(%)	CPU Time(s)	ARPD(%)	CPU Time(s)
10	0.32	< 1	0.32	< 1	0.31	< 1	16.92	< 1	16.98	< 1
50	0.23	< 1	0.22	< 1	0.29	< 1	14.05	< 1	14.11	< 1
100	0.10	1.77	0.10	1.80	0.26	1.31	13.56	< 1	13.59	< 1
150	0.11	4.86	0.11	4.85	0.26	6.24	10.98	4.75	10.99	1.44
200	0.09	10.50	0.09	10.01	0.25	20.40	10.41	14.96	10.44	3.82
300	0.13	28.81	0.14	29.33	0.26	96.80	13.45	64.99	13.46	21.47
Average	0.16		0.16		0.27		13.23		13.26	

problem under study of allocating jobs into a variable number of groups is NP-hard. Based on some obtained properties, we proposed two batch-based heuristics NBA and UBA. Even though they always provide worse solutions than the enumeration method IEM, they are much faster. IEM is impractical for large instances. After observing that the effectiveness of NBA and UBA deteriorates with problem size, we presented two iterated greedy algorithms which are more effective than the adapted existing PTA, New_FF and New_BF for large sized problems (New_FF and New_BF are effective for periodical cases). The proposed IG algorithms spend acceptable computational time even for large size problems.

In the future, single machine group scheduling problems in cloud computing and services computing are promising topics.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (No. 2017YFB1400801), the National Natural Science Foundation of China (Nos. 61572127, 61872077, 61832004) and Collaborative Innovation Center of Wireless Communications Technology. Rubén Ruiz is supported by the Spanish Ministry of Economy and Competitiveness, under the project "SCHEYARD-Optimization of scheduling problems in container yards" (No. DPI2015-65895-R) partly financed with FEDER funds.

References

[1] S.-Y. Wang and L. Wang, "An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 1, pp. 139–149, 2016.

[2] Q.-K. Pan, "An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling," *European Journal of Operational Research*, vol. 250, no. 3, pp. 702–714, 2016.

[3] X. Li, L. Gao, and W. Li, "Application of game theory based hybrid algorithm for multi-objective integrated process planning and scheduling," *Expert Systems with Applications*, vol. 39, no. 1, pp. 288–297, 2012.

[4] D. Biskup, "Single-machine scheduling with learning considerations," *European Journal of Operational Research*, vol. 115, no. 1, pp. 173–178, 1999.

[5] S. Teyarachakul, S. Chand, and J. Ward, "Effect of learning and forgetting on batch sizes," *Production and Operations Management*, vol. 20, no. 1, pp. 116–128, 2011.

[6] W. Smith, I. Foster, and V. Taylor, "Predicting application run times using historical information," in *Job Scheduling Strategies for Parallel Processing*. Springer, 1998, pp. 122–142.

[7] S.-J. Yang, D.-L. Yang, and T.-R. Chang, "Single-machine scheduling with joint deteriorating and learning effects under group technology and group availability assumptions," *Journal of the Chinese Institute of Industrial Engineers*, vol. 28, no. 8, pp. 597–605, 2011.

[8] S.-J. Yang, "Group scheduling problems with simultaneous considerations of learning and deteriorating effects on a single-machine," *Applied Mathematical Modelling*, vol. 35, no. 8, pp. 4008–4016, 2011.

[9] Y. Yin, W.-H. Wu, T. Cheng, and C.-C. Wu, "Single-machine scheduling with time-dependent and position-dependent deteriorating jobs," *International Journal of Computer Integrated Manufacturing*, vol. 28, no. 7, pp. 781–790, 2015.

[10] Y. He and L. Sun, "One-machine scheduling problems with deteriorating jobs and position-dependent learning effects under group technology considerations," *International Journal of Systems Science*, vol. 46, no. 7, pp. 1319–1326, 2015.

[11] X. Huang, M.-Z. Wang, and J.-B. Wang, "Single-machine group scheduling with both learning effects and deteriorating jobs," *Computers and Industrial Engineering*, vol. 60, no. 4, pp. 750–754, 2011.

[12] D. Biskup, "A state-of-the-art review on scheduling with learning effects," *European Journal of Operational Research*, vol. 188, no. 2, pp. 315–329, 2008.

[13] X. Li, Y. Jiang, and R. Ruiz, "Methods for scheduling problems considering experience, learning, and forgetting effects," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 5, pp. 743–754, 2018.

[14] Y. Wang, X. Li, and R. Ruiz, "An exact algorithm for the shortest path problem with position-based learning effects," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 11, pp. 3037–3049, 2017.

[15] Y. Yeh, C. Low, and W.-Y. Lin, "Single machine scheduling with time-dependent learning effect and non-linear past-sequence-dependent setup times," *Journal of Applied Mathematics and Physics*, vol. 3, no. 1, p. 10, 2015.

[16] C.-H. Wu, W.-C. Lee, P.-J. Lai, and J.-Y. Wang, "Some single-machine scheduling problems with elapsed-time-based and position-based learning and forgetting effects," *Discrete Optimization*, vol. 19, pp. 1–11, 2016.

[17] M. Ji, D. Yao, Q. Yang, and T. Cheng, "Single-machine common flow allowance scheduling with aging effect, resource allocation, and a rate-modifying activity," *International Transactions in Operational Research*, vol. 22, no. 6, pp. 997–1015, 2015.

[18] T. Keshavarz, M. Savelsbergh, and N. Salmasi, "A branch-and-bound algorithm for the single machine sequence-dependent group scheduling problem with earliness and tardiness penalties," *Applied Mathematical Modelling*, vol. 39, no. 20, pp. 6410–6424, 2015.

[19] Y.-T. Xu, Y. Zhang, and X. Huang, "Single-machine ready times scheduling with group technology and proportional linear deterioration," *Applied Mathematical Modelling*, vol. 38, no. 1, pp. 384–391, 2014.

[20] K. Rustogi and V. A. Strusevich, "Single machine scheduling with general positional deterioration and rate-modifying maintenance," *Omega*, vol. 40, no. 6, pp. 791–804, 2012.

[21] P. Perez-Gonzalez and J. Framinan, "Single machine scheduling with

periodic machine availability," *Computers and Industrial Engineering*, vol. 123, pp. 180–188, 2018.

[22] E. Pan, G. Wang, L. Xi, L. Chen, and X. Han, "Single-machine group scheduling problem considering learning, forgetting effects and preventive maintenance," *International Journal of Production Research*, vol. 52, no. 19, pp. 5690–5704, 2014.

[23] M. Liu, S. Wang, C. Chu, and F. Chu, "An improved exact algorithm for single-machine scheduling to minimize the number of tardy jobs with periodic maintenance," *International Journal of Production Research*, vol. 54, no. 12, pp. 3591–3602, 2016.

[24] M. Iranpoor, S. Ghomi, and M. Zandieh, "Machine scheduling in the presence of sequence-dependent setup times and a rate-modifying activity," *International Journal of Production Research*, vol. 50, no. 24, pp. 7401–7414, 2012.

[25] X. Zhang, Y. Yin, and C.-C. Wu, "Scheduling with non-decreasing deterioration jobs and variable maintenance activities on a single machine," *Engineering Optimization*, no. 1, pp. 84–97, 2017.

[26] H. Liu, A. Gong, and M. Hu, "An optimal control method for fuzzy supplier switching problem," *International Journal of Machine Learning and Cybernetics*, vol. 6, no. 4, pp. 651–654.

[27] W. Liu, S.-J. Song and C. Wu, "The loss-averse newsvendor problem with random yield," *Transactions of the Institute of Measurement & Control*, vol. 36, no. 3, pp. 312-320,2014.

[28] W.-H. Kuo and D.-L. Yang, "Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect," *European Journal of Operational Research*, vol. 174, no. 2, pp. 1184–1190, 2006.

[29] R. K. Otnes, L. Enochson, and M. Maqusi, *Applied Time Series Analysis, Vol. 1*. Academic Press,, 1978.

[30] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, pp. 287–326, 1979.

[31] D. Li and X. Sun, *Nonlinear integer programming*. Springer Science & Business Media, 2006, vol. 84.

[32] G. H. Hardy and E. M. Wright, *An introduction to the theory of numbers*. Oxford University Press, 1979.

[33] Y.-Y. Lu, J.-J. Wang, and J.-B. Wang, "Single machine group scheduling with decreasing time-dependent processing times subject to release dates," *Applied Mathematics and Computation*, vol. 234, pp. 286–292, 2014.

[34] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.

[35] ——, "An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives," *European Journal of Operational Research*, vol. 187, no. 3, pp. 1143–1159, 2008.

[36] Q.-K. Pan and R. Ruiz, "An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem," *Omega*, vol. 44, pp. 41–50, 2014.

[37] I. Ribas, R. Companys, and X. Tort-Martorell, "An iterated greedy algorithm for the flowshop scheduling problem with blocking," *Omega*, vol. 39, no. 3, pp. 293–301, 2011.

[38] F. J. Rodriguez, M. Lozano, C. Blum, and C. García-Martínez, "An iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem," *Computers & Operations Research*, vol. 40, no. 7, pp. 1829–1841, 2013.

Haiyan Xu received her B.Sc. degree at the School of Mathematical Science from Nanjing Normal University in 2001 and her M.Sc. degrees at the Department of Mathematics from Southeast University in 2009 respectively. She is currently a Ph.D. candidate at the School of Computer Science and Engineering, Southeast University, Nanjing, China. Her main interests focus on Task Scheduling.

Xiaoping Li (M'09-SM'12) received his B.Sc. and M.Sc. degrees in Applied Computer Science from the Harbin University of Science and Technology, Harbin, China, in 1993 and 1999 respectively. He obtained his Ph.D. degree in Applied Computer Science from the Harbin Institute of Technology, Harbin, China, in 2002. He joined Southeast University, Nanjing, China, in 2005, and is currently a professor at the School of Computer Science and Engineering. From Jan. 2003 to Dec. 2004, he did postdoctoral research at the Department of Automation at Tsinghua University, Beijing, China. From Mar. 2009 to Mar. 2010, he was a visiting professor at the National Research Council, London, Ontario, Canada. He is the author or co-author of more than 100 academic papers, some of which have been published in international journals such as *IEEE Transactions on Automation Science and Engineering*, *IEEE Transactions on Cybernetics*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Systems, Man and Cybernetics: Systems*, *IEEE Transactions on Cloud Computing*, *Information Sciences*, *Omega*, *European Journal of Operational Research*, *International Journal of Production Research*, *Future Generation Computer Systems*, *Journal of Grid Computing*, *Journal of Supercomputing*, *Expert Systems with Applications* and *Journal of Network and Computer Applications*. His research interests focus on Scheduling in Cloud Computing, Scheduling in Cloud Manufacturing, Machine Scheduling, Project Scheduling and Terminal Container Scheduling.

Rubén Ruiz received B.Sc. and M.Sc. degrees in computer science engineering from the Universitat Politècnica de València in 1998 and 2000, respectively, and the Ph.D. degree in statistics and operations research from the same university in 2003. He is a full professor of Statistics and Operations Research at the Universitat Politècnica de València, Spain. He is co-author of more than 60 papers in International Journals and has participated in presentations of more than a hundred and fifty papers in national and international conferences. He is editor of the Elsevier's journal *Operations Research Perspectives (ORP)* and co-editor of the JCR-listed journal *European Journal of Industrial Engineering (EJIE)*. He is also associate editor of other important journals like *TOP* or *Applied Mathematics and Computation* as well as a member of the editorial boards of several journals most notably *European Journal of Operational Research* and *Computers and Operations Research*. He is the director of the Applied Optimization Systems Group (SOA, http://soa.iti.es) at the Instituto Tecnológico de Informática (ITI, http://www.iti.es) where he has been principal investigator on several public research projects as well as privately funded projects with industrial companies. His research interests include scheduling and routing in real life scenarios.

Haihong Zhu received her B.Sc. degree at the School of Computer and Information Engineering from Chuzhou University, Chuzhou, China, in 2011. She obtained the M.Sc. degree at School of Computer and Information from Anhui Normal University, Wuhu, China, in 2015. She is currently pursuing the Ph.D degree at the School of Cyber Science and Engineering, Southeast University, Nanjing, China. Her current research interests include algorithm design and scheduling optimization.