ALGORITHMS FOR SINGLE-MACHINE SCHEDULING PROBLEM WITH DETERIORATION DEPENDING ON A NOVEL MODEL

HUA-PING WU
Accounting R&D Center, Chongqing University of Technology
Chongqing 400054, China

MIN HUANG
College of Information Science and Engineering, Northeastern University
State Key Laboratory of Integrated Automation of Process Industry
(Northeastern University)
Shenyang 110004, China

W. H. Ip
Department of Industrial and Systems Engineering
The Hong Kong Polytechnic University
Hong Kong 110004, China

QUN-LIN FAN
College of Management, Chongqing University of Technology
Chongqing 400054, China

(Communicated by Eugene Levner)

ABSTRACT. In this paper, a novel single machine scheduling problem with deterioration depending on waiting times is investigated. Firstly, a new deterioration model for the problem is presented. Secondly, according to characteristics of the problem, dominance properties and lower bounds are proposed and integrated into the Branch and Bound algorithm (B&B) to solve the small-medium scale problems. Thirdly, for solving a large-scale problem, the Rules Guided Nested Partitions method (RGNP) is proposed. The numerical experiments show that when the size of the problem is no more than 17 jobs, the B&B algorithm can obtain the optimal solutions in a reasonable time. The RGNP method can also obtain an average error percentage for near-optimal solutions less than 0.048 within 0.2s. The analysis shows the efficiency of RGNP, and, hence, it can be used for solving large size problems.

1. Introduction. In recent literature on scheduling problems with deterioration, the actual processing times $p_j$ of job $j$ is assumed to be dependent on the job starting time $s_j$, which means the job always deteriorates from the time 0 to the starting time, i.e., $p_j = a_j + bs_j$, where $b$ is deterioration rate. However, in some realistic problems, this assumption is not appropriate. For example, in steel production, heating the ingots is necessary to soften the ingots for rolling and in providing a
sufficiently high initial temperature to ensure that the rolling processing is completed in the required temperature region. The temperate of the ingots, however, while waiting to enter the rolling machine, drops below a certain level. This phenomenon is known as deterioration by Gupta and Gupta (1998) [7]. In such a case, if we assume that the ingots deteriorate depending on starting time, it is not reasonable. Because the starting time is from the time of entering heating furnace to rolling, however, actually, the time of ingots deterioration is from the time of outing from heating furnace to rolling. Therefore, the time of ingots deterioration should be from the time of outing from heating furnace (i.e., release time) to rolling (i.e., starting time). Based on above reasons, in this paper, the deterioration model of jobs is defined as a function of waiting times, i.e., \( p_j = a_j + b(s_j - r_j) \), where \( r_j \) is the release time of job \( j \). From the above two mathematical formulation of the actual processing time, it is clear seen that the new model is more appropriate to reflect the real problem, but it also increases the difficulty of solving the problem.

The deterioration job scheduling problem was first introduced independently by Gupta and Gupta (1998) [7] and by Browne and Yechiali (1990) [1]. Since then, related models have been extensively studied from a variety of perspectives. For instance, Cheng et al. (2010) [3] considered a new deterioration model where the actual processing time of a job was a function of the normal processing times of the jobs already processed and they proved that the problems of minimizing the total weighted completion time, maximum lateness, and maximum tardiness were polynomially solvable. Lai and Lee (2010) [9] studied a new deterioration model in which the actual job processing time was a function of the original processing times of jobs already processed and the scheduled positions. They showed that certain objectives, including makespan, total completion time, total weighted completion time, total tardiness and maximum tardiness under the proposed model, remained polynomially solvable. Shen et al. (2013) [17] considered the single-machine scheduling problem with time-dependent deterioration and gave an optimal solution corresponding to the particular problem. Ruiz-Torres et al. (2013) [16] introduced the deteriorating effect into parallel machine scheduling and considered the objective of makespan minimization. Qian and Steiner (2013) [15] introduced fast algorithms for solving single machine scheduling problems with learning/deterioration effects and time-dependent processing times, with or without due date assignment. Yin et al. (2015) [23] introduced the actual processing time of a job depends on the starting time and scheduled position of the job and showed that the related problems can be solved in \( O(n \log n) \). Oron (2014) [13] gave a polynomial time solution for the problem of a single-machine setting under the assumptions of general linear deterioration and convex resource functions. Kacem and Levner (2016) [8] propose a new dynamic programming algorithm and a faster fully polynomial time approximation scheme for the problem of scheduling a set of proportional deteriorating non-resumable jobs on a single machine subject to maintenance.

The literature mentioned above focuses on the fact that the jobs are always available. Since some practical problems need to consider different release times, for example, in steel production, different ingots have different release times. In this paper, we propose a novel single machine scheduling problem with deterioration depending on waiting times.

The main contribution of this paper is that a novel single machine scheduling problem with deterioration depending on waiting times is presented. Corresponding to the problem, a new deterioration model is proposed, and in order to solve the
problem, a Branch and Bound algorithm (B&B) integrating with dominance properties and lower bounds is proposed for solving small-medium scale problems; for solving larger scale problems, the Rules Guided Nested Partitions method (RGNP) is proposed. The results of numerical experiments show that for a size of no more than 17 jobs, the B&B algorithm can obtain the optimal solution in a reasonable time. The RGNP method can obtain the average error percentage of near-optimal solutions less than 0.048 within 0.2s. The analysis indicates the efficiency of RGNP, such that it can be used for solving large scale problems.

The rest of this paper is organized as follows. In Section 2, the problem is formulated. The branch and bound with dominance properties and low bounds and the rules guided nested partitions method are proposed for solving the problem studied in Section 3. The numerical experimentation is described in Section 4, followed by the conclusions in the last section.

2. Problem formulation. This paper considers the single machine scheduling problem with deterioration depending on waiting times to minimize the makespan.

This problem can be described as follows. Assume that there are $n$ jobs that are required to be scheduled. The normal processing time and release time for job $j$ ($j = 1, 2, \ldots, n$) are $a_j$ and $r_j$, respectively. All jobs have a common deterioration rate $b$ ($b > 0$). The actual processing time $p_j$ of job $j$ is a function of $a_j$, $b$ and the waiting time $s_j - r_j$, i.e., $p_j = a_j + b(s_j - r_j)$, where $s_j$ is the starting time of job $j$. The objective is to obtain an optimal schedule to minimize the makespan $C_{\text{max}}$. This problem can be denoted as $1|p_j = a_j + b(s_j - r_j), r_j|C_{\text{max}}$ by using the three-field notation scheme $\alpha|\beta|\gamma$ introduced by Graham et al. (1979)\[2\]. Since Cheng and Ding (1998)\[2\] have proved that the makespan problem with identical deteriorating jobs $1|p_j = a_j + bs_j, r_j|C_{\text{max}}$ is strongly NP-complete, clearly, the problem $1|p_j = a_j + b(s_j - r_j), r_j|C_{\text{max}}$ is also strongly NP-complete.

3. The B&B and RGNP method. In this section, the branch and bound algorithm integrating with the dominance properties and lower bounds, and the rules guided nested partitions method are proposed to solve the single machine scheduling problem with arbitrary deterioration rates and release times. Firstly, dominance properties and lower bounds are proposed in subsection 3.1 and 3.2, respectively. Then, the B&B and RGNP method are described in detail in subsection 3.3 and 3.4, respectively.

3.1. Dominance properties. Assume that schedules $S = (\pi, i, j, \pi')$ and $S' = (\pi, i, j, \pi')$. Here, the schedule $S = \{\pi, i, j, \pi'\}$ is a sequence, in which $\pi$ is regarded to be scheduled sequence with $k - 1$ jobs, $i$ in the $k^{th}$ position and $j$ in the $(k + 1)^{th}$ position to be unscheduled jobs, and $\pi'$ to be unscheduled sequence with $n - (k + 1)$ jobs. For convenience, generally, we only need to consider that the effect of $i$ and $j$ are scheduled at different locations. Let $t$ be the completion time of the last job in $\pi'$. Several properties are given as follows.

Property 1. If the release times of all jobs are identical, then there is an optimal schedule by ordering jobs with non-decreasing of $a_j$.

Property 2. If $t < r_i < r_j$ and $r_i + a_i < r_j$, then there is an optimal schedule with job $i$ before job $j$.

Property 3. If $t \leq \min\{r_i, r_j\}$ and $2r_i + a_i < 2r_j + a_j$, then there is an optimal schedule with job $i$ before job $j$. 

All above properties can be proved by the internal adjacent exchange method. Here, they are omitted.

3.2. Lower bounds. In this subsection, three lower bounds are developed for minimizing the makespan problem with arbitrary deterioration rates and release times.

Assume that $S$ denotes the set of scheduled jobs which includes $k$ jobs, $C[k]$ the completion time of the $k^{th}$ job in $S$, $US$ the set of unscheduled jobs, and $f^*$ the optimal makespan. Specially, $C[0] = 0$.

**Proposition 1.** $LB_1 = (1 + b)^{n-k}C[k] + \sum_{j=k+1}^{n} (1 + b)^{n-j}a_j - \sum_{i=k+1}^{n} b(1 + b)^{n-i}r[i]$, with by ordering the normal processing times $a[j]$ of jobs in $US$ according to the short processing time rule and by the release times $r[i]$ of jobs in $US$ according to the late release time rule being a lower bound.

**Proof.** The scheduled set $S$ includes $k$ jobs, the completion time of the $(k + 1)^{th}$ job is:

$$C[k+1] = C[k] + a[k+1] + b(\max\{C[k], r[k+1]\} - r[k+1]) \geq C[k] + a[k+1] + b(C[k] - r[k+1]) = (1 + b)C[k] + a[k+1] - br[k+1].$$

The completion time of the $(k + 2)^{th}$ job is:

$$C[k+2] = C[k+1] + a[k+2] + b(\max\{C[k+1], r[k+2]\} - r[k+2]) \geq C[k+1] + a[k+2] + b(C[k+1] - r[k+2]) = (1 + b)^2C[k] + (1 + b)a[k+1] + a[k+2] - b(1 + b)r[k+1] - br[k+2].$$

Similarly, the completion time of the $n^{th}$ job is:

$$C[n] \geq (1 + b)^{n-k}C[k] + \sum_{j=k+1}^{n} (1 + b)^{n-j}a_j - \sum_{i=k+1}^{n} b(1 + b)^{n-i}r[i]$$

i.e., $f^* \geq C[n] \geq (1 + b)^{n-k}C[k] + \sum_{j=k+1}^{n} (1 + b)^{n-j}a_j - \sum_{i=k+1}^{n} b(1 + b)^{n-i}r[i]$. 

Set $\phi = (1 + b)^{n-k}C[k] + \sum_{j=k+1}^{n} (1 + b)^{n-j}a_j - \sum_{i=k+1}^{n} b(1 + b)^{n-i}r[i]$, the lower bound of the partial sequence $S$ is larger than or equal to the minimum value of $\phi$. Clearly, it is observed that the first term of $\phi$ is known, so it is only needed to minimize the second term and maximize the third term of $\phi$. Since $(1 + b)^{n-j}$ gradually decreases as $j$ increases, the minimum value of $\phi$ can be obtained by ordering the normal processing times of jobs in $US$ according to the short processing time rule and by the release times of jobs in $US$ according to the late release time rule. So: $LB_1 = \min\{(1 + b)^{n-k}C[k] + \sum_{j=k+1}^{n} (1 + b)^{n-j}a_j - \sum_{i=k+1}^{n} b(1 + b)^{n-i}a[i]\}$. 

**Proposition 2.** $LB_2 = \max_{j \in US}(r_j + a_j)$ is a lower bound.

**Proof.** In an optimal schedule, each job $j$ in $US$ will not start before its release time $r_j$ and will complete its processing no later than $f^*$, So it is processed during the interval of time $[r_j, f^*]$. In this interval of time, all jobs of $US$ are processed on the machine. So: $f^* \geq \max_{j \in US}(r_j + a_j)$, i.e., $LB_2 = \max_{j \in US}(r_j + a_j)$. 

Proposition 3. \( LB_3 = r_j + a_j + b(C_{[k]} - r_j)^+; j \in \{ j \mid r_j = \max_{j \in US} r_j \} \) is a lower bound.

Proof. In an optimal schedule, the job \( j \) with the latest release time of \( US \) will start after its release time \( r_j \) and complete its processing before the optimal makespan \( f^* \). So:

\[
\begin{align*}
    f^* & \geq \max_{j \in US} \{ r_j + p_j \} \\
    & \geq r_j + p_j \quad \quad \quad j \in \{ j \mid r_j = \max_{j \in US} r_j \} \\
    & \geq r_j + a_j + b(C_{[k]} - r_j)^+ \quad j \in \{ j \mid r_j = \max_{j \in US} r_j \}
\end{align*}
\]

i.e., \( LB_3 = r_j + a_j + b(C_{[k]} - r_j)^+; j \in \{ j \mid r_j = \max_{j \in US} r_j \} \). \( \Box \)

In summary, in order to obtain a tight lower bound, select the largest value among \( LB_1 \), \( LB_2 \) and \( LB_3 \) as a lower bound, i.e., \( LB = \max_{b=1,2,3} LB_b \).

3.3. The branch and bound algorithm. As a classic algorithm, B&B is just a simple search algorithm with framework. If it is directly applied to the problem in this paper, the CPU time will be unacceptable when the number of jobs is 7. The contribution of this paper is that we not only use the depth first search strategy but combine with corresponding properties and low bounds proposed in subsection 3.1 and 3.2.

The branch and bound algorithm is firstly proposed by Land and Doig (1960) [10] and by Dakin (1965) [5]. It is also successfully applied to many areas, for example, cyclic scheduling in a robotic cell [22], rescheduling considering limited disturbance [21] and so on. In this paper, the branch and bound algorithm mainly uses the backtracking method, which incorporates systematicity with jumping characteristics. It adopts a depth first search strategy to make the search start from the root node to the whole solution space. When the algorithm searches any node in the solution space tree, it needs to judge whether the sub-tree of the node as a root contains solutions of the problem or not. If not, it will jump over all sub-trees of the node as a root and then backtrack to its father node step by step. Otherwise, it continues to search the sub-tree. If a whole sequence is obtained and its objective value is less than the current one, it will then replace the current one. Moreover, since the backtracking method only records a current sequence and its lower bounds, it makes the store space become small to a great extent. In this paper, we adopt the depth first search in the branch and bound procedure, respectively. Dominance properties and lower bounds are used for eliminating the nodes that do not satisfy the solutions of the problem. In the following paragraph, the branch and bound algorithm is described in more detail.

The branch and bound algorithm includes several following elements.

Node
A search tree consists of many nodes, each of which denotes a partial schedule.

Branch
To branch is to generate all children nodes by the current active node. Each child node denotes a branch.
Search strategy with eliminate nodes
Search strategy is designed according to the depth first search.

Step 1 Generate all children nodes of the current expansion node.
Step 2 In all children nodes, eliminate the nodes which cannot obtain the optimal schedule according to properties 1-3.
Step 3 Add the remaining nodes in all children nodes into a list of active nodes.
Step 4 Select a node from a list of active nodes as the next expansion node, which is expanded until the maximum depth is reached.

Repeat the above steps until no more active nodes can be expanded.

Upper bound and lower bound
At the beginning of the algorithm, calculate the makespan of sequences obtained according to the job’s current minimum completion time \((1 + b) \max(r_j, t) + a_j - br_j\), and select it as the initial upper bound, which will be replaced with the better solution that is generated in the search procedure.

Lower bound, \(LB\), is used for eliminating the nodes which cannot develop the optimal solution. If the new node cannot be eliminated by the properties 1-3, its \(LB\) will be calculated.

Backtracking
When all child nodes of the current node have been searched, the algorithm will backtrack to the father node of the current node and continue to search other nodes of the father node.

The procedure of B&B algorithm can be summarized as follows.

Step 1 Initialization
Calculate the initial upper bound, go to Step 2.

Step 2 Branching
All child nodes are generated by the current active node, go to Step 3.

Step 3 Search strategy
The recently generated node is selected as an active node, which is firstly expanded. Apply the properties 1-3 to eliminate child nodes of the expanded node which cannot develop the optimal solution, go to Step 4.

Step 4 Lower bound
Calculate the lower bound for each remaining node. If it is less than the current optimal solution, continue to search its branches. If it is equal to the current optimal solution, go to Step 6. Otherwise, eliminate it, continue to search other child nodes of the current active node. When a whole sequence is obtained, its makespan replaces the current optimal solution. Go to Step 5.

Step 5 Backtracking
Backtrack the father node of the current node and continue to search other children nodes of the father node. If no more nodes can be searched, go to Step 6. Otherwise, Go to Step 2.

Step 6 Stopping
Output an optimal solution.

3.4. Rules guided nested partitions method (RGNP). The original nested partitions method (ONP) was first developed by Shi and Olafsson (2000) for global optimization, and firstly proposed for solving discrete problems. It has good convergence, and integrates the capacity of the local search with that of a global search. The Nested Partitions method has been applied to knowledge discovery in databases, the local pickup and delivery problem and the discrete facility
The main idea is that the feasible region will be continuously partitioned until the most promising region includes a singleton solution, i.e., at the beginning of the method. The whole solution space is then considered as the most promising region $\sigma(0)$ and partitioned into several sub-regions and the depth of the method $d$ is 0. Then, through sampling, select a sub-region with the best promising index as the most promising region in the next step. Generally, the promising index is defined as the value of the objective. At each iteration, other than the first step, if a sub-region is selected as the most promising region $\sigma(d)$, then other regions except the most promising region are aggregated into one region which is called surrounding region or complementary region $\phi$. Generally, the method mainly comprises four elements, i.e., partitioning, sampling, selection and backtracking, the detailed process of which can be referred by Shi and Olafsson (2000) [18].

Based on original nested partitions method, a solution approach named RGNP is proposed in the section for solving the NP-hard problem of a single machine with deterioration depending on waiting times, where its objective is to minimize the makespan. Specially, a number of rules are incorporated into the sampling procedure to promote algorithm convergence, whereas in the surrounding region, randomly sampling is used to ensure the diversities of solutions and avoids trapping into the local optimal solutions. The RGNP method also has four important elements, partitioning and stopping, sampling, selection and backtracking, which are described in the following.

Give a set $N = \{1, 2, ..., n\}$, all permutations of $\{1, 2, ..., n\}$ constitute the whole solution space. In depth 0, the whole solution space is considered as the most promising region $\sigma(0)$. It is divided into $n$ sub-regions by fixing the first job on the machine to be one of 1, 2, ..., and $n$. The current most promising region can be divided into $n - d$ sub-regions when the depth is $d$. The algorithm won’t stop until the most promising region contains only a singleton solution. This process is shown as Figure 1, where $j_{k}^{S}$ denotes the $k^{th}$ job in schedule $S$. Moreover, if the algorithm always backtracks, the algorithm will stop when the times of the algorithm backtracks to the whole region are more than 100.

**Sampling**

To find the most promising region for the next depth, three related rules are given and used for sub-regions.

Rule 1 Obtain randomly a partial sequence from the unscheduled jobs.

Rule 2 Obtain a partial sequence by ordering jobs in non-decreasing order of the smallest $(1 + b) \max(r_j, t) + a_j - br_j$ from the unscheduled jobs, where $t$ is the completion time of the last jobs scheduled, especially, $t = 0$ when the depth is 0.

Rule 3 Obtain a partial sequence according to the short normal processing time from unscheduled jobs.

The above partial sequences combining with the scheduled partial sequence constitute three samples. These three samples are solutions of the problem in the sub-regions.

For the surrounding region, randomly sampling is used.

**Selection**
The most promising region

First, the makespan of each sample from the sampling procedure is calculated and the best makespan is chosen as the promising index of each sub-region. The best makespan in all sub-regions and the surrounding region is defined as $R^*$. Then, the most promising region corresponding to $R^*$ for the next step is determined by the following cases:

Case 1 If $R^*$ corresponds to the sub-region of the current most promising region, then it will be partitioned in the next step.

Case 2 If $R^*$ corresponds to more than one sub-region of the current most promising region, then one of them is selected randomly and partitioned in the next step.

Case 3 If $R^*$ accords with the surrounding region, then we adopt backtracking, the procedure of which will be introduced in later.

Case 4 If $R^*$ corresponds to the surrounding region and one or more than one of the current most promising region, then one of them is selected randomly with identical probability. In this case, if one of the current most promising regions is selected, it will be the most promising region to be partitioned in the next step. On the contrary, backtracking will be used when the surrounding region is selected.

**Backtracking**

If the surrounding region accords with the current most promising region, then the method backtracks to the adjacent super-region of the current most promising region. If the method always backtracks to the adjacent super-region of the same current most promising region, then it will backtrack to the whole solutions region when the times are more than 100.

**Figure 1.** The generic partitions for the single machine scheduling problem
The procedure of the RGNP method can be summarized as follows.

Step 1 Initialization
Set the overall solution space as the initial most promising region and the initial surrounding region as $\varphi$. Go to Step 2.

Step 2 Partition & Stopping
If the current most promising region is a singleton solution region, then the method will stop and the best solution obtained is returned. Otherwise, the current most promising region is partitioned into several sub-regions. Go to Step 3.

Step 3 Sampling
Obtain samples from each of sub-regions according to three rules and a random sample from the surrounding region. Calculate the promise indices for both the several sub-regions and the surrounding region. Go to Step 4.

Step 4 Selection
Select the most promising region among sub-regions and the surrounding region. If the surrounding region is selected, go to Step 5. Otherwise, go to Step 2.

Step 5 Backtracking
The method backtracks to the adjacent super-region. If the times of backtracking to the adjacent super-region of the same current most promising region are more than 100, then the method will backtrack to the whole region. Go to Step 1. If the times of backtracking to the whole region for the same current most promising region are more than 100, then the method will stop, and the best solution obtained from the sample of the surrounding region is returned.

4. Numerical experimentation. In this section, the numerical experimentation is used to evaluate the performance of the three algorithms. The experimental design follows the framework used by Chu (1992)[4]. All programming is run on the same personal computer with Intel (R) Core (TM) 2 processors. The normal processing times of the jobs are generated from a uniform distribution on $(1,100)$. The release times are generated from a uniform distribution on $(0, 50\lambda)$ where $n$ is the size of jobs and $\lambda$ is a control variable, which decides the scatter range of the release dates. The B&B, RGNP and ONP are used for solving the problem. The following experiments are tested.

Firstly, the performance of the B&B, RGNP and ONP with respect to a parameter $\lambda$ is tested.

The size of jobs is fixed at 8, the deterioration rate $b = 0.05, 0.1$, and the control variable $\lambda$ takes values from 0.2 to 3.0 with an increment of 0.2 each time. For each $b$ and $\lambda$, 100 replications are randomly generated. When $b = 0.05$, the average error rates of RGNP and ONP are recorded in Figure 2, and the average CPU time of algorithms is recorded in Figure 3. Likewise, when $b = 0.1$, the average error rates of RGNP and ONP are recorded as in Figure 4, and the average CPU times of the algorithms are recorded as in Figure 5.

From Figures 2 and 4, when the deterioration rate $b = 0.05, 0.01$ and the control variable $\lambda$ takes the values from 0.2 to 3.0 with an increment of 0.2 each time, the average error rate of RGNP is near or even 0 in most cases. Compared with RGNP, the average error rate of ONP is not stable and the largest value is more than 0.2. From Figure 3 and 5, when the deterioration rate $b = 0.05, 0.1$, and the control variable $\lambda$ takes the values from 0.2 to 3.0 with an increment of 0.2 each time, the
average CPU time of RGNP and ONP is less than 1s. The average CPU time of the B&B is increasing with increasing in the whole view. The average CPU time of the B&B for $b = 0.05$ is more than that in case of $b = 0.1$ when $\lambda = 3.0$.

Secondly, the performance of the B&B, RGNP and ONP with respect to a parameter $b$ is tested. The size of jobs is fixed at 8, the control variable $\lambda = 0.2, 3.0$, and the deterioration rate $b$ takes the values from 0.025 to 0.5 with an increment of 0.025 each time. For each $\lambda$ and $b$, 100 replications are randomly generated. When $b = 0.05$, average error rates of RGNP and ONP are recorded as shown in Figure 6, and the average CPU times of the algorithms are recorded in Figure 7. Likewise, when $b = 0.1$, average error rates of RGNP and ONP are as shown in Figure 8, and the average CPU time of the algorithms is shown in Figure 9.

From Figures 6 and 8, when the control variable and the deterioration rate takes values from 0.025 to 0.5 in steps of 0.025 each time, the average error rate of RGNP is 0 in most cases. In Figure 6, when $\lambda = 0.2$, the average error rate of ONP is also not stable and the largest is more than 0.2. In Figure 8, when $\lambda = 3.0$ the average error rate of ONP is increasing with $b$ increasing. As seen in Figure 7, the average CPU time of the B&B is no more than 2.5s when $\lambda = 0.2$. However, when $\lambda = 3.0$, it will need much longer time as in Figure 9, especially, when $b = 0.05$, it needs more than 30s.

Therefore, from testing $\lambda$ and $b$, a few of conclusions are as follows:
The control variable $\lambda$ takes values of 0.2, 1.0 and 3.0, and the deterioration rate $b$ takes 0.05 and 0.1. Four different sizes of jobs ($n = 5, 9, 13, 17$) are adopted. The average error rate of RGNP and ONP, respectively, relative to the optimal solution, are provided in Table 1. The error rate is $\text{var} = (H - H^*) / H^* \times 100\%$, where $H$ is a solution from RGNP or ONP, and $H^*$ is an optimal solution from the B&B. The running time of the B&B, RGNP and ONP are also recorded.

According to Table 1, the results are as follows:

(i) The branch and bound algorithm can obtain the optimal solution within 11 hours when the job size is equal to or smaller than 17.
(ii) The RGNP can obtain solutions which have the average error percentage less than 0.048 in no more than 0.2s.
Table 1. Comparison of results among algorithms

| n  | b  | λ   | Average error rate | Average CPU time(s) |
|----|----|-----|--------------------|---------------------|
|    |    |     | RGNP   | ONP    | RGNP  | ONP  |
| 5  | 0.05 | 0.2  | 0.000  | 0.000  | 1.07  | 0.00  | 0.00 |
|    | 1.0  | 0.034 | 0.054 | 0.94  | 1.26  | 0.00  | 0.00 |
|    | 3.0  | 0.000 | 0.000  | 1.17  | 0.00  | 0.00  | 0.00 |
| 0.10 | 0.2  | 0.000  | 0.004  | 1.36  | 0.00  | 0.00  | 0.00 |
|    | 1.0  | 0.000 | 0.000  | 0.82  | 0.00  | 0.00  | 0.00 |
|    | 3.0  | 0.000 | 0.005  | 1.10  | 0.00  | 0.00  | 0.00 |
| Average CPU time |  |  |  |  |  | 1.10  | 0.00  | 0.00 |
| 9  | 0.05 | 0.2  | 0.000  | 0.002  | 1.50  | 0.01  | 0.00 |
|    | 1.0  | 0.010 | 0.106 | 114.30 | 0.01  | 0.00  | 0.00 |
|    | 3.0  | 0.000 | 0.012  | 1.38  | 0.01  | 0.00  | 0.00 |
| 0.10 | 0.2  | 0.000  | 0.015  | 1.07  | 0.00  | 0.00  | 0.00 |
|    | 1.0  | 0.000 | 0.070  | 11.04 | 0.01  | 0.00  | 0.00 |
|    | 3.0  | 0.016 | 0.071  | 21.53 | 0.01  | 0.00  | 0.00 |
| Average CPU time |  |  |  |  |  | 1315.55 | 0.04  | 0.02 |
| 13 | 0.05 | 0.2  | 0.008  | 0.028  | 16.88 | 0.04  | 0.02 |
|    | 1.0  | 0.016 | 0.300  | 51.19 | 0.05  | 0.02  | 0.02 |
|    | 3.0  | 0.000 | 0.031  | 6041.71 | 0.06  | 0.01  | 0.01 |
| 0.10 | 0.2  | 0.028  | 0.057  | 13.16 | 0.04  | 0.02  | 0.02 |
|    | 1.0  | 0.021 | 0.221  | 35.32 | 0.05  | 0.02  | 0.02 |
|    | 3.0  | 0.000 | 0.109  | 1735.06 | 0.04  | 0.01  | 0.01 |
| Average CPU time |  |  |  |  |  | 25.16  | 0.01  | 0.00 |
| 17 | 0.05 | 0.2  | 0.018  | 0.038  | 1007.11 | 0.18  | 0.07 |
|    | 1.0  | 0.015 | 0.200  | 1222.28 | 0.18  | 0.07  | 0.07 |
|    | 3.0  | 0.012 | 0.216  | 38012.51 | 0.18  | 0.07  | 0.07 |
| 0.10 | 0.2  | 0.048  | 0.116  | 610.34 | 0.12  | 0.07  | 0.07 |
|    | 1.0  | 0.022 | 0.444  | 1958.67 | 0.12  | 0.07  | 0.07 |
|    | 3.0  | 0.012 | 0.245  | 18292.45 | 0.12  | 0.07  | 0.07 |
| Average CPU time |  |  |  |  |  | 10183.89 | 0.15  | 0.07 |

Figure 10. The CPU time of several algorithms with respect to n

(iii) The average error percentage of the ONP becomes larger when the size increases.
Moreover, the average CPU time of several algorithms with respect to the job size $n$ is shown in Figure 10, the time of ONP and RGNP are very few, but the time of B&B increases sharply when the job size increases.

It is shown that when the job size is equal to or smaller than 17, the B&B can be used for obtaining the optimal solution within 11 hours.

In order to further verify the efficiency of RGNP, the following experiments are done. The control variable $\lambda$ takes values of 0.2, 1.0 and 3.0, and the deterioration rate $b$ takes 0.05 and 0.1. Six different sizes of jobs ($n = 20, 40, 60, 80, 100, 120$) are adopted. The average error rate (AER) of RGNP and ONP, respectively, relative to the optimal solution, are provided in Table 2. The error rate is $\text{var} = (H - H^*)/H^* \times 100\%$, where $H$ is a solution from RGNP or ONP, and $H^*$ is a best solution between RGNP and ONP. The average CPU time (Time) of the RGNP and ONP are recorded.

**Table 2. Comparison of results among algorithms**

| $n$ | $b$ | $\lambda$ | AER | Time(s) |
|-----|-----|-----------|-----|---------|
|     |     |           | RGNP | ONP     | RGNP | ONP |
| 20  | 0.05| 0.2       | 0.723| 0.009  | 0.06 | 0.05 |
|     | 1.0 | 0.000     | 0.014| 0.009  | 0.07 | 0.05 |
|     | 0.1 | 1.0       | 0.721| 0.009  | 0.06 | 0.05 |
|     | 0.1 | 0.000     | 0.009| 0.009  | 0.07 | 0.05 |
|     | 3.0 | 0.000     | 0.723| 0.009  | 0.06 | 0.05 |
| 40  | 0.05| 0.2       | 0.723| 0.009  | 0.07 | 0.05 |
|     | 1.0 | 0.000     | 0.014| 0.009  | 0.06 | 0.05 |
|     | 0.1 | 1.0       | 0.721| 0.009  | 0.07 | 0.05 |
|     | 0.1 | 0.000     | 0.009| 0.009  | 0.07 | 0.05 |
|     | 3.0 | 0.000     | 0.723| 0.009  | 0.06 | 0.05 |
| 60  | 0.05| 0.2       | 0.723| 0.009  | 0.07 | 0.05 |
|     | 1.0 | 0.000     | 0.014| 0.009  | 0.06 | 0.05 |
|     | 0.1 | 1.0       | 0.721| 0.009  | 0.07 | 0.05 |
|     | 0.1 | 0.000     | 0.009| 0.009  | 0.07 | 0.05 |
|     | 3.0 | 0.000     | 0.723| 0.009  | 0.06 | 0.05 |
| 80  | 0.05| 0.2       | 0.723| 0.009  | 0.07 | 0.05 |
|     | 1.0 | 0.000     | 0.014| 0.009  | 0.06 | 0.05 |
|     | 0.1 | 1.0       | 0.721| 0.009  | 0.07 | 0.05 |
|     | 0.1 | 0.000     | 0.009| 0.009  | 0.07 | 0.05 |
|     | 3.0 | 0.000     | 0.723| 0.009  | 0.06 | 0.05 |
| 100 | 0.05| 0.2       | 0.723| 0.009  | 0.07 | 0.05 |
|     | 1.0 | 0.000     | 0.014| 0.009  | 0.06 | 0.05 |
|     | 0.1 | 1.0       | 0.721| 0.009  | 0.07 | 0.05 |
|     | 0.1 | 0.000     | 0.009| 0.009  | 0.07 | 0.05 |
|     | 3.0 | 0.000     | 0.723| 0.009  | 0.06 | 0.05 |
| 120 | 0.05| 0.2       | 0.723| 0.009  | 0.07 | 0.05 |
|     | 1.0 | 0.000     | 0.014| 0.009  | 0.06 | 0.05 |
|     | 0.1 | 1.0       | 0.721| 0.009  | 0.07 | 0.05 |
|     | 0.1 | 0.000     | 0.009| 0.009  | 0.07 | 0.05 |
|     | 3.0 | 0.000     | 0.723| 0.009  | 0.06 | 0.05 |
From Table 2, the performance of RGNP is far better than that of ONP. The maximum CPU time of RGNP is only 76.76s when \( n = 120 \) and the solution of it is far superior to that of ONP. For ONP, it obtains solutions which become inferior over the job size increase. The analysis further shows the efficiency of the RGNP, hence, the RGNP can be regarded as a good method for solving large scale problems.

5. Conclusions. In this paper, a novel single machine scheduling problem with deterioration depending on waiting times is presented. Firstly, a new deterioration model corresponding to the problem is proposed. Then, the branch and bound algorithm integrating with the dominance properties and lower bounds is proposed to obtain optimal solutions of small-medium scale problems. Since the branch and bound has a limit when applied to large scale problems, the rules guided nested partitions method is proposed. The results of the numerical examples with a job size smaller than 18 jobs show that, the B&B algorithm can obtain the optimal solutions in a reasonable time, and the RGNP method can obtain near-optimal solutions, with an average error rate of less than 0.048 in no more than 0.2s. When the number of jobs becomes large, RGNP can still obtain good near-optimal solutions within a short time. According to the results of the analysis, it shows the efficiency of RGNP. Therefore, it can be used for solving large size problems. In the future, multiple machines scheduling problems and other good algorithms (such as approximate algorithms\[24\], tabu search algorithm\[11\] and so on) will be considered.

Acknowledgments. The authors would like to thank the anonymous referees and the editor for improving the exposition of this paper.

This work is supported by the National Science Foundation for Distinguished Young Scholars of China under Grant No. 71325002.; Major International Joint Research Project of NSFC under Grant No. 71620107003; the Foundation for Innovative Research Groups of National Science Foundation of China under Grant No. 61621004; the 111 Project (B16009); the Fundamental Research Funds for State Key Laboratory of Synthetical Automation for Process Industries under Grant No.2013ZCX11;the National Social Science Foundation of China under Grant No. 16CGL070; the Humanities and Social Science Research General Project of Chongqing Education Committee in 2015 under Grant No. 15SKG135; the Social Science Planning Doctor Project of Chongqing under Grant No. 14YJC630032; the Chongqing Foundation and Advanced Research Project under Grant No. cste2013jcyjA0022; the Scientific Research Foundation of Chongqing University of Technology under Grant No.20014Z007. The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China(PolyU 152108/15E).

REFERENCES

[1] S. Browne and U. Yechiali, Scheduling deteriorating jobs on a single processor, *Operations Research*, 38 (1990), 495–498.

[2] T. C. E. Cheng and Q. Ding, The complexity of scheduling starting time dependent tasks with release times, *Information Processing Letters*, 65 (1998), 75–79.

[3] T. C. E. Cheng, W. C. Lee and C. C. Wu, Single-machine scheduling with deteriorating functions for job processing times, *Applied Mathematical Modelling*, 34 (2010), 4171–4178.

[4] C. B. Chu, A branch-and-bound algorithm to minimize total flow time with unequal release dates, *Naval Research Logistics*, 39 (1992), 859–875.
[5] R. J. Dakin, A tree-search algorithm for mixed integer programming problem, *The Computer Journal*, 8 (1965), 250–256.

[6] R. L. Graham, E. L. Lawler and J. K. Lenstra, Optimalization and approximation in the deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics*, 5 (1979), 287–326.

[7] J. N. D. Gupta and S. K. Gupta, Single facility scheduling with nonlinear processing times, *Computers and Industrial Engineering*, 14 (1998), 387–393.

[8] I. Kacem and E. Leven, An Improved approximation scheme for scheduling a maintenance and proportional deteriorating jobs, *Journal of Industrial and Management Optimization*, 12 (2016), 811–817.

[9] P. J. Lai and W. C. Lee, Single-machine scheduling with a nonlinear deterioration function, *Information Processing Letters*, 110 (2010), 455–459.

[10] A. H. Land and A. G. Doig, An automatic method of solving discrete programming problems, *Econometrica*, 28 (1960), 497–520.

[11] Y. K. Lin and C. S. Chong, A tabu search algorithm to minimize total weighted tardiness for the job shop scheduling problem, *Journal of Industrial and Management Optimization*, 12 (2016), 703–717.

[12] S. Olafsson and J. Yang, Intelligent partitioning for feature selection, *INFORMS Journal on Computing*, 17 (2005), 339–355.

[13] D. Oron, Scheduling controllable processing time jobs in a deteriorating environment, *Journal of the Operational Research Society*, 1 (2014), 49–56.

[14] L. Pi, Y. Pan and L. Shi, Hybrid nested partitions and mathematical programming approach and its applications, *IEEE Transactions on Automation Science & Engineering*, 5 (2008), 573–586.

[15] J. Qian and G. Steiner, Fast algorithms for scheduling with learning effects and time-dependent processing times on a single machine, *European Journal of Operational Research*, 225 (2013), 547–551.

[16] A. J. Ruiz-Torres, G. Paletta and E. Pérez, Parallel machine scheduling to minimize the makespan with sequence dependent deteriorating effects, *Computers & Operations Research*, 40 (2013), 2051–2061.

[17] P. Shen, C. M. Wei and X. Huang, Single-machine scheduling problems with an actual time-dependent deterioration, *Applied Mathematical Modelling*, 37 (2013), 5555–5562.

[18] L. Shi and S. Olafsson, Nested partitions method for global optimization, *Operations Research*, 48 (2000), 390–407.

[19] L. Shi, S. Olafsson and Q. Chen, An optimization framework for product design, *Management Science*, 47 (2001), 1681–1692.

[20] S. A. Shihabi and S. Olafsson, A hybrid of nested partition, binary ant system, and linear programming for the multidimensional knapsack problem, *Computers & Operations Research*, 37 (2010), 247–255.

[21] P. Yan, A. Che, X. Cai and X. Tang, Two-phase branch and bound algorithm for robotic cells rescheduling considering limited disturbance, *Name of the Journal*, 50 (2014), 128–140.

[22] P. Yan, C. Chu, N. Yang and A. Che, A branch and bound algorithm for optimal cyclic scheduling in a robotic cell with processing time windows, *International Journal of Production Research*, 48 (2010), 6461–6480.

[23] Y. Yin, W. H. Wu, T. C. E. Cheng and C. C. Wu, Single-machine scheduling with time-dependent and position-dependent deteriorating jobs, *International Journal of Computer Integrated Manufacturing*, 28 (2015), 781–790.

[24] X. Zhang, D. Xu, D. Du and C. Xiao, Approximate algorithms for unrelated machine scheduling to minimize makespan, *Journal of Industrial and Management Optimization*, 12 (2016), 771–779.

Received March 2014; 1st revision February 2015; final revision January 2016.

E-mail address: wuhuaping2010@126.com
E-mail address: mhuang@neu.edu.cn
E-mail address: mfwhip@polyu.edu.hk
E-mail address: fqunlin@126.com