Improved Guided Nested Partitions Method for Single-Machine Scheduling Problem with Deterioration Depending on Piece-wise Function

Hua-Ping Wu, *Member, IAENG,* Min Huang and Qun-Lin Fan

***Abstract*—In order to accord with the actual scheduling problems, this paper presents a new single machine scheduling problem. The main contribution in the paper is that a novel deterioration model is proposed firstly and integrated into the scheduling problem, and three algorithms are presented to solve the problems. Firstly, according to the characteristics of steel production, a new deterioration model depending on piece-wise function 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 to solve the small-medium scale problems. Thirdly, for solving a large-scale problem, the Improved Guided Nested Partitions method and heuristic algorithm based on minimum completion time are proposed. The numerical experiments show that for the medium-small scale problem, the Branch and Bound algorithm can obtain the optimal solutions in a reasonable time. Heuristic algorithm can also obtain good near-optimal solutions. The Improved Guided Nested Partitions method is prior to heuristic algorithm since it can obtain the average error percentage of near-optimal solutions less than 0.036 within 0.2s. The analysis shows the efficiency of the Improved Guided Nested Partitions method. Therefore, the Improved Guided Nested Partitions method and heuristic algorithm can be used for solving large size problems.**

***Index Terms*—Single machine, Piece-wise function, Branch and Bound algorithm, Improved Guided Nested Partitions.**

I. Introduction

In recent literatures on scheduling problems with deterioration, the actual processing times of jobs are assumed to be dependent on the job starting time. However, this assumption is not appropriate for many realistic problems. 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(Gupta and Gupta[1]). In such a case, the ingots will require reheating, and thus increases the processing time. And in the process of deteriorating, the temperature of ingots will not change when it drops to normal temperature. Moreover, ingots will not start deteriorating before the release time. Clearly, if the deterioration model depends on the starting time, it is not reasonable. Therefore, in this paper, the deterioration model of jobs is defined as apiece-wise function.

The deterioration job scheduling problem was first introduced independently by Gupta and Gupta[1] and by Browne and Yechiali[2]. Since then, related models have been extensively studied from a variety of perspectives and seen from Cheng et al.[3], Lai and Lee[4], Shen et al.[5], Ruiz-Toreset al.[6], Qian and Steiner[7], Wang et al.[8] and Yin et al.[9].

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 a deterioration model according to the realistic case.

The main contribution of this paper is that a novel single machine scheduling problem with deterioration depending on a piece-wise function is presented. 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 (IGNP) and heuristic algorithm based on minimum completion time (HAMC) are proposed. The results of numerical experiments show that for a size of no more than 15 jobs, the B&B algorithm can obtain the optimal solution in a reasonable time. The IGNP is prior to HAMC since it can obtain the average error percentage of near-optimal solutions less than 0.036 within 0.2s, and HAMC can also obtain good near-optimal solutions. The analysis indicates the efficiency of IGNP, 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, the Improved Guided Nested Partitions method and the heuristic algorithm based on minimum completion time 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.

This work was supported by the National Science Foundation for Distinguished Young Scholars of China under Grant No. 71325002, No. 61225012; Major International Joint Research Project of NSFC under Grant No.71620107003; the National Social Science Foundation of China under Grant No.16CGL070; the Fundamental Research Funds for State Key Laboratory of Synthetical Automation for Process Industries under Grant No.2013ZCX11; the Humanity and Social Science Youth foundation of Ministry of Education of China under Grant No.14YJC630032; Chongqing Foundation and Advanced Research Project under Grant No. cstc2013jcyjA90022; the Humanities and Social Science Research General Project of Chongqing Education Committee under Grant No.15SKG135, No.16SKGH135; Chongqing social science planning and cultivating project under Grant No.yjg153039; the Spark Program of Youth Research Program of Chongqing University of Technology; and the Scientific Research Foundation of Chongqing University of Technology under Grant No.20014Z007.

H.P. Wu is with Accounting R&D Center, Chongqing University of Technology, Chongqing, 400054, China, e-mail: hpwu@cqut.edu.cn.

M. Huang is with School of Information Science and Engineering, Northeastern University. Q.L. Fan is with School of Management, Chongqing University of Technology.

II. Problem Formulation

In the existed literatures, the actual processing times of ingots in steel production are assumed to be dependent on the job starting time. However, this assumption is not appropriate for many realistic problems. 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. In such a case, the ingots will require reheating, and thus increases the processing time. And in the process of deteriorating, the temperature of ingots will not change when it drops to normal temperature. Moreover, ingots will not start deteriorating before the release time. Clearly, if the deterioration model depends on the starting time, it is not reasonable. Therefore, in this paper, the deterioration model of jobs is defined as a piece-wise function which depends on the waiting time or the period from ingots out of heating furnace to the temperature of them dropping to normal temperature.

Based on the above problems, this paper considers the single machine scheduling problem with deterioration depending on a piece-wise function to minimize the makespan. This problem can be described as follows.

Assume that there are n jobs 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 own common deterioration rate $b(b > 0)$. The actual processing time p_j of job j is a piece-wise linear function of a_j, b and the waiting time, i.e.,

$$p_j = \begin{cases} a_j + bw_j, & s_j - r_j < W; \\ a_j + bW, & s_j - r_j \geq W. \end{cases} \quad (1)$$

where $w_j = s_j - r_j$, s_j is the starting time of job j and W a constant. If the waiting time of a job is larger than or equal to W, the actual processing time of the job is $a_j + bW$. The objective is to obtain an optimal schedule to minimize the makespan $C_{\max}$. This problem can be denoted as $1 \left| p_j = \begin{cases} a_j + bw_j; \\ a_j + bW. \end{cases}, r_j \right| C_{\max}$ by using the three-field notation scheme $\alpha \,|\beta\,|\gamma$ introduced by Graham et al.[10]. Since Cheng and Ding[11] have proved that the makespan problem with identical deteriorating jobs $1 \,|p_j = a_j + bs_j, r_j\,| C_{\max}$ is strongly NP-complete. Clearly, the problem $1 \left| p_j = \begin{cases} a_j + bw_j; \\ a_j + bW. \end{cases}, r_j \right| C_{\max}$ is also strongly NP-complete.

III. The B&B, IGNP Method and HAMC

The main contribution of the section is that dominance properties and lower bounds are proposed, the rule of searching of B&B is designed, IGNP and HAMC are presented 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 branch and bound algorithm integrating with the dominance properties and lower bounds is given in subsection 3.3. Finally, the Improved Guided Nested Partitions method and heuristic algorithm based on minimum completion time are described in detail in subsection 3.4 and 3.5, respectively.

A. Dominance Properties

Assume that schedules $S = (\pi, i, j, \pi')$ and $S' = (\pi, j, i, \pi')$, where π with job i in the k^{th} position and π' are partial sequences. Let t be the completion time of the last job in π. 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 $r_j < r_i < t$ and $t - r_i < W < t - r_j$, then there is an optimal schedule with job i before job j.

Property 4 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. Specially, when $W < (r_i - r_j) + a_i$, then job i and j has an identical rank.

Property 5 If $t \geq \max\{r_i, r_j\}$, $t - r_j > W$ and $t + a_j - r_i < (1 - b)W$, 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.

B. 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 $LB_1 = \max\limits_{j \in US}(r_j + a_j)$ is a lower bound.

Proof. In an optimal schedule, each job j of US will not start before its release time r_j and will complete its processing no later than f^*. So:

$$f^* \geq \max_{j \in US}(r_j + a_j)$$

i.e., $LB_1 = \max\limits_{j \in US}(r_j + a_j)$.

Proposition 2 $LB_2 = r_j + a_j, j \in \{j \left| r_j = \max\limits_{j \in US} r_j \right.\}$ is a lower bound.

Proof. In an optimal schedule, each job j of US will not start before its release time r_{j} and will complete its processing no later than f^*. So:

$$f^* \geq r_j + a_j, j \in \{j \left| r_j = \max_{j \in US} r_j \right.\}$$

i.e., $LB_2 = r_j + a_j, j \in \{j \left| r_j = \max\limits_{j \in US} r_j \right.\}$.

Proposition 3 $LB_3 = \min\limits_{j \in US} r_j + \sum\limits_{j \in US} a_j$ is a lower bound.

Proof. In an optimal schedule, each job j of US will not start before its release time $r_{\min}$ and will complete its processing no later than f^*, so it is processed during the interval of time $[r_{\min}, f^*]$. In this interval of time, all jobs of US are processed on the machine. So:

$$f^* \geq \min_{j \in US} r_j + \sum_{j \in US} p_j \geq \min_{j \in US} r_j + \sum_{j \in US} a_j$$

i.e., $LB_3 = \min\limits_{j \in US} r_j + \sum\limits_{j \in US} a_j$.

Proposition 3 $LB_4 = C_{[k]} + \sum_{j \in US} a_j$ is a lower bound.

Proof. In an optimal schedule, each job j of US will not start before its release time $r_{\min}$ and will complete its processing no later than f^*, so it is processed during the interval of time $[C_{[k]}, f^*]$. In this interval of time, all jobs of US are processed on the machine. The actual processing times of jobs is larger than or equal to $\min(C_{[k]} - r_j, W)$ when the release times of jobs is less than $C_{[k]}$, So:

$f^* \geq C_{[k]} + \sum_{j \in \{r_j | C_{[k]} > r_j\}} b \min(C_{[k]} - r_j, W) + \sum_{j \in US} p_j$

$\geq C_{[k]} + \sum_{j \in US} a_j$ i.e., $LB_4 = C_{[k]} + \sum_{j \in US} a_j$.

In summary, in order to obtain a tight lower bound, select the largest value among LB_1, LB_2, LB_3 and LB_4 as a lower bound, i.e., $LB = \max_{b=1,2,3,4} LB_b$.

C. The Branch and Bound Algorithm

In this paper, the branch and bound algorithm mainly uses the backtracking method. It 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 1-5.

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 short processing time and early release time rules, and select the smallest one 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 1-5, 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.

Procedure of B&B algorithm

Step1. 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 1-5 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.

D. Improved Guided Nested Partitions Method

The scheduling problems have been solved using other algorithms[12][13][14], in the section, we also use the another proper algorithm to solve the scheduling problem. Shi and Olafsson[15] first developed original nested partitions method for solving discrete problems. It has been applied to many fields(Olafsson and Yang[16], Pi et al.[17], Shi et al.[18], Shihabi and Olafsson[19]). Motivated by its success in other applications, and the problem considered in this study being atypical discrete problem, the improved guided nested partition method (IGNP) is proposed for solving the single machine scheduling problem $1 \left| p_j = \begin{cases} a_j + bw_j; \\ a_j + bW. \end{cases}, r_j \right| C_{\max}$.

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 φ. Generally, the method mainly comprises four elements, i.e., partitioning, sampling, selection and backtracking(Shi and Olafsson[15]).

In this subsection, we propose the improved guided nested partition method for solving the NP-hard problem of a single machine with deterioration depending on waiting times, where its objective is to minimize the makespan. It also has four important elements, partitioning and stopping, sampling, selection and backtracking, which are described in the following.

Partitioning and stopping

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 $1, 2, ..., andn$. The current most promising region can be divided into $n-d$ sub-regions when the depth is d . The algorithm wont stop until the most promising region contains only a singleton solution. 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, four related sampling methods are given and used for each sub-region.

M1. Obtain randomly a partial sequence from the unscheduled jobs.

M2. 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.

M3. Obtain a partial sequence according to the short normal processing time from unscheduled jobs.

M4. Obtain a partial sequence according to the early release time from unscheduled jobs.

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

For the surrounding region, randomly sampling is used to ensure the diversities of solutions and avoids trapping into the local optimal solutions.

Selection

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 situation:

S1. If R^* corresponds to the sub-region of the current most promising region, then it will be partitioned in the next step.

S2. 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.

S3. If R^* accords with the surrounding region, then we adopt backtracking, the procedure of which will be introduced in later.

S4. 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.

The procedure of the the improved guided nested partition 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 ϕ. 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-region. Go to Step 3.

Step 3. Sampling

Obtain samples from each sub-region according to three sampling methods 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.

E. Heuristic Algorithm based on Minimum Completion Time

According to the characteristics of the problem, the heuristic algorithm based on minimum completion time (HAMC) is presented in this section. The procedure of it is as follows.

Step 1: Parameter Initialization. Assume that $k = 1$, $i = 1$, $r_{\max} = \max_{j=1}^{n} r_j$ and $J = \{1, 2, \ldots, n\}$. $C_{[k]}$ is the completion time of the k^{th} job. Specially, $C_{[0]} = 0$;

Step 2: If $C_{[k-1]} < r_{\max}$, then select the job J_j form J with the smallest $\max(r_j, C_{[k-1]}) + a_j + b\min(W, \max(r_j, C_{[k-1]}) - r_j)$, and assign it in the k^{th} position, let $C_{[k]} = \max(r_j, C_{[k-1]}) + a_j + b\min(W, \max(r_j, C_{[k-1]}) - r_j)$, and delete J_j form J, go to Step 4; otherwise, go to Step 3.

Step 3: Select the job from J with the smallest a_j and assign it in the k^{th} position, and delete J_j form J, let $C_{[k]} = \max(r_j, C_{[k-1]}) + a_j + b\min(W, \max(r_j, C_{[k-1]}) - r_j)$, go to Step 4;

Step 4: If $k < n$, let $k = k + 1$, go to Step 2;

Step 5: The algorithm ends. And output a current optimal solution.

IV. 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[20]. 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

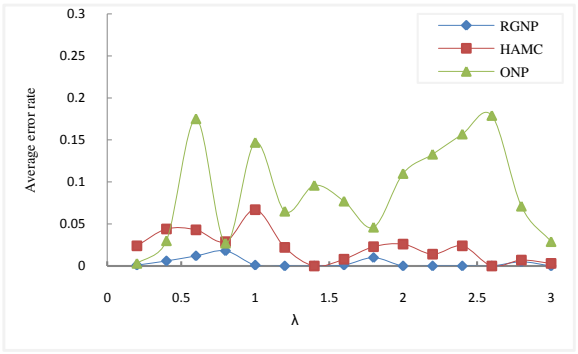

Fig. 1. The effect of λ on average error rate based on $b = 0.05$

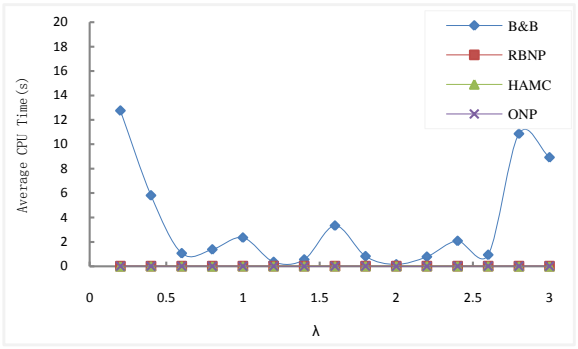

Fig. 2. The effect of λ on average CPU time based on $b = 0.05$

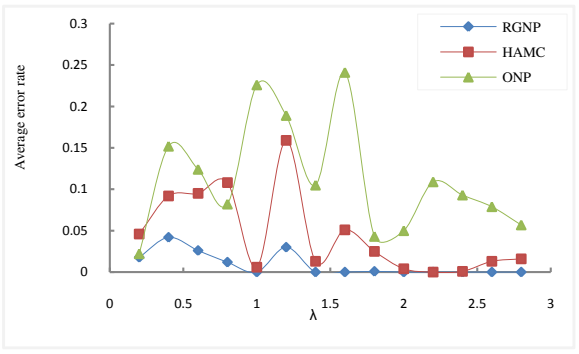

Fig. 3. The effect of λ on average error rate based on $b = 0.1$

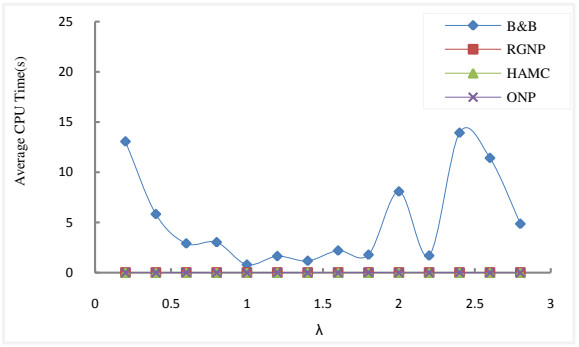

Fig. 4. The effect of λ on average CPU time based on $b = 0.1$

on (1,100). The release times are generated from a uniform distribution on $(0, 50.5n\lambda)$ where n is the size of jobs and λ is a control variable, which decides the scatter range of the release dates. The maximum waiting time $W = \sum_{j=1}^{n} a_j$. The B&B, IGNP, HAMC and ONP are used for solving the problem. The following experiments are tested.

Firstly, the performance of the B&B, IGNP, HAMC and ONP with respect to a parameter λ is tested.

The size of jobs is fixed at 6, the deterioration rate $b = 0.05, 0.1$, and the control variable λ takes values from 0.2 to 3.0 with an increment of 0.2 each time. For each b and λ, , 100 replications are randomly generated. When $b = 0.05$, the average error rates of IGNP, HAMC and ONP are recorded in Figure 1, and the average CPU time of algorithms is recorded in Figure 2. Likewise, when $b = 0.1$, the average error rates of IGNP, HAMC and ONP are recorded as in Figure 3, and the average CPU times of the algorithms are recorded as in Figure 4.

From Figures 1 and 3, when the deterioration rate $b = 0.05, 0.1$ and the control variable λ takes the values from 0.2 to 3.0 with an increment of 0.2 each time, the average error rate of IGNP is near or even 0 in most cases. Compared with IGNP, the average error rate of ONP is not stable, and that of HAMC is between of them.

From Figure 2 and 4, when the deterioration rate $b = 0.05, 0.1$ and the control variable λ takes the values from 0.2 to 3.0 with an increment of 0.2 each time, the average CPU time of IGNP, HAMC and ONP is less than 1s. The average CPU time of the B&B is increasing when λ becomes larger or smaller. The average CPU time of the B&B is less than 10s when λ takes the values from 0.4 to 2.0.

Secondly, the performance of the B&B, IGNP, HAMC and ONP with respect to a parameter b is tested.

The size of jobs is fixed at 6, 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 λ and b, 100 replications are randomly generated. When $b = 0.05$, average error rates of IGNP, HAMC and ONP are recorded as shown in Figure 5, and the average CPU times of the algorithms are recorded in Figure 6. Likewise, when $b = 0.1$, average error rates of IGNP, HAMC and ONP are as shown in Figure 7, and the average CPU time of the algorithms is shown in Figure 8.

From Figures 5 and 7, when the control variable $\lambda = 0.2, 3.0$ and the deterioration rate b takes values from 0.025 to 0.5 in steps of 0.025 each time, the average error rate of IGNP is 0 in most cases. The average error rate of ONP is also not stable. And the average error rate of HAMC is between of them. From Figures 6 and 8, the average CPU

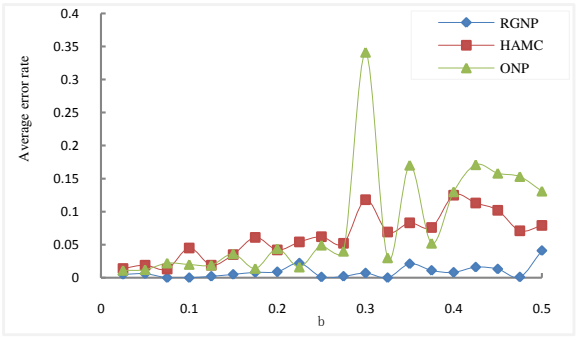

Fig. 5. The effect of b on average error rate based on $\lambda = 0.2$

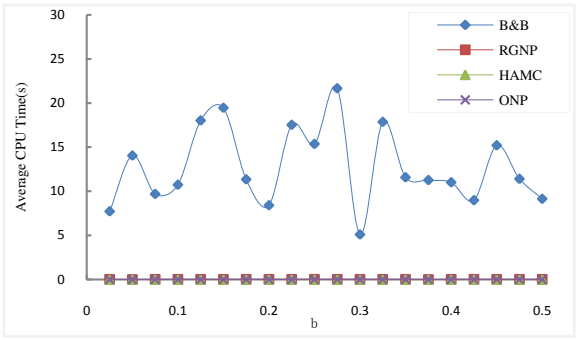

Fig. 6. The effect of b on average CPU time based on $\lambda = 0.2$

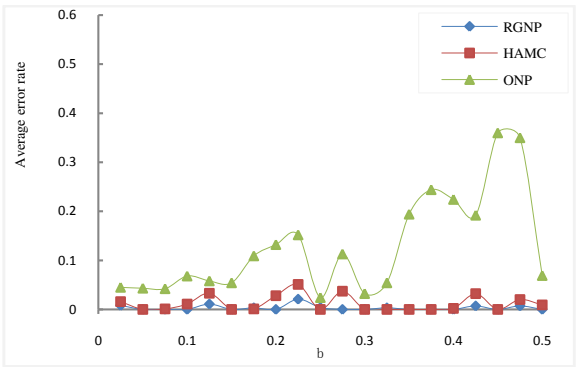

Fig. 7. The effect of b on average error rate based on $\lambda = 3.0$

times of IGNP, HAMC, ONP are less than 1s. For B&B, it is seen from Figure 6 that the average CPU time of the B&B is stable and no more than 25s when $\lambda = 0.2$. But from Figure 8, the average CPU time of the B&B is increasing with the value of b decreasing when $\lambda = 3.0$. For example, it will need more than 40s when $b = 0.025$ and $\lambda = 3.0$.

Therefore, by testing λ and b, a few of conclusions are as follows:

1) when the release times are more scattered and the deterioration rate is less, the B&B will need much time to obtain the optimal solution;

2) the performance of IGNP is hardly affected by λ and b;

3) the performance of HAMC is between IGNP and ONP, and it is hardly affected by λ and b;

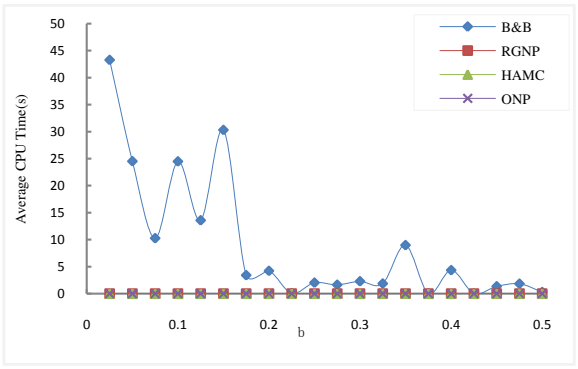

Fig. 8. The effect of b on average CPU time based on $\lambda = 3.0$

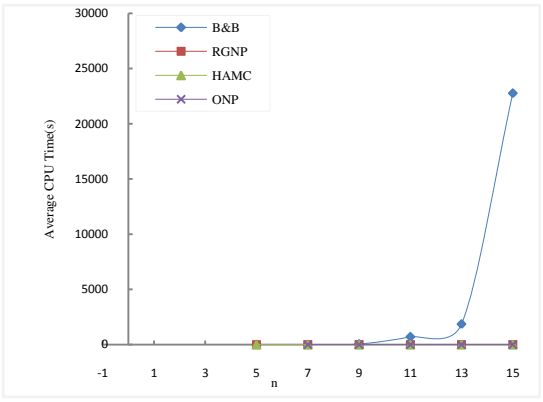

Fig. 9. The effect of b on average CPU time based on $\lambda = 3.0$

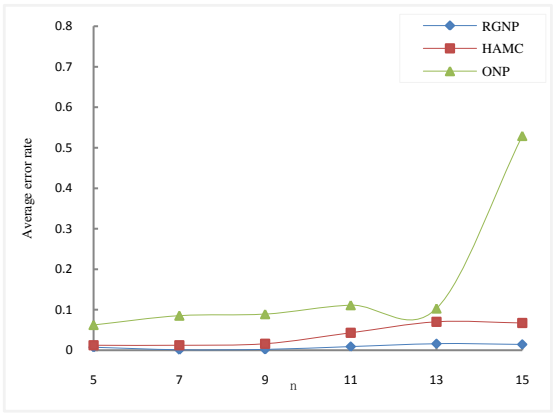

Fig. 10. The effect of b on average CPU time based on $\lambda = 3.0$

4) the time performance of ONP is affected by λ and b, But the average error rate of it is not stable because of the change of λ and b.

Finally, in order to further verify the B&B, IGNP, HAMC and ONP validity, the following experiments are tested.

(1) The performance of the B&B, IGNP, HAMC and ONP for solving the small scale problem is tested.

The control variable λ takes values of 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 $(n = 5, 7, 9, 11, 13, 15)$ are adopted. The average error rate of IGNP, HAMC and ONP, respectively, relative to the optimal solution, are provided in Table 1. The error rate is $var = (H - H^*)/H^* \times 100\%$, where H is a solution from IGNP, HAMC or ONP, and H^* is an optimal solution from the B&B. The average CPU time of the B&B, IGNP, HAMC and ONP are recorded. And the average number of nodes of the B&B is also recorded.

Moreover, the average CPU time of algorithms with respect to the job size n is shown in Figure 9. It is seen that: the average CPU times of ONP, HAMC and IGNP are slowly increasing with the size increase, but the time of B&B increases sharply when the job size increases.

TABLE I
COMPARISON OF ALGORITHMS BASED ON OPTIMAL SOLUTIONS

				Average error rate			Average CPU time(s)			
n	b	λ	nodes	IGNP	HAMC	ONP	B&B	IGNP	HAMC	ONP
5	0.05	0.2	80	0.030	0.039	0.031	1.143	0.003	0.004	0.001
		1.0	46	0.000	0.001	0.254	0.348	0.004	0.001	0.002
		3.0	69	0.000	0.000	0.000	0.421	0.003	0.003	0.002
	0.10	0.2	131	0.012	0.022	0.013	1.921	0.003	0.001	0.002
		1.0	55	0.000	0.000	0.033	0.602	0.003	0.003	0.002
		3.0	37	0.000	0.010	0.041	0.067	0.003	0.004	0.002
		Average		0.007	0.012	0.062	0.750	0.003	0.002	0.002
7	0.05	0.2	2250	0.002	0.012	0.015	6.215	0.014	0.002	0.002
		1.0	121	0.000	0.007	0.075	0.208	0.012	0.005	0.003
		3.0	1490	0.000	0.000	0.013	6.415	0.013	0.003	0.003
	0.10	0.2	3695	0.008	0.047	0.012	6.706	0.009	0.005	0.008
		1.0	70	0.000	0.000	0.291	0.027	0.011	0.005	0.003
		3.0	763	0.000	0.009	0.106	3.190	0.008	0.006	0.008
		Average		0.001	0.012	0.085	3.358	0.011	0.004	0.004
9	0.05	0.2	164701	0.006	0.023	0.008	180.447	0.028	0.006	0.003
		1.0	1356	0.000	0.000	0.175	1.090	0.031	0.006	0.003
		3.0	4313	0.000	0.002	0.127	3.082	0.031	0.006	0.003
	0.10	0.2	9682	0.000	0.059	0.004	8.802	0.031	0.006	0.003
		1.0	399	0.004	0.012	0.187	2.068	0.016	0.016	0.015
		3.0	169557	0.000	0.000	0.037	160.362	0.016	0.015	0.015
		Average		0.002	0.016	0.089	59.308	0.025	0.009	0.007
11	0.05	0.2	370784	0.000	0.022	0.027	1309.532	0.032	0.015	0.013
		1.0	65210	0.011	0.062	0.189	290.845	0.044	0.016	0.013
		3.0	254639	0.006	0.011	0.046	318.612	0.045	0.015	0.016
	0.10	0.2	63471	0.016	0.073	0.036	185.452	0.047	0.016	0.015
		1.0	409939	0.021	0.093	0.373	159.736	0.044	0.015	0.016
		3.0	88025	0.000	0.000	0.000	1116.147	0.047	0.016	0.16
		Average		0.009	0.043	0.111	713.387	0.043	0.015	0.038
13	0.05	0.2	3956334	0.033	0.039	0.044	4330.128	0.047	0.016	0.015
		1.0	175396	0.007	0.084	0.141	300.670	0.078	0.015	0.032
		3.0	2372912	0.002	0.071	0.132	1100.308	0.078	0.015	0.015
	0.10	0.2	787922	0.007	0.083	0.035	1463.735	0.078	0.031	0.016
		1.0	6590	0.044	0.091	0.124	210.570	0.078	0.016	0.031
		3.0	258226	0.003	0.053	0.143	3780.062	0.078	0.016	0.015
		Average		0.016	0.070	0.103	1864.246	0.072	0.018	0.020
15	0.05	0.2	53250447	0.003	0.052	0.322	19272.428	0.125	0.035	0.031
		1.0	32685	0.012	0.131	0.600	1620.129	0.130	0.046	0.031
		3.0	10988782	0.021	0.089	0.704	9377.331	0.125	0.032	0.030
	0.10	0.2	10258684	0.008	0.091	0.891	90089.424	0.125	0.032	0.031
		1.0	252483	0.036	0.021	0.483	2680.127	0.130	0.046	0.047
		3.0	31243455	0.002	0.021	0.178	13535.881	0.125	0.046	0.032
		Average		0.014	0.067	0.529	22762.55	0.126	0.039	0.033

TABLE II
COMPARISON OF ALGORITHMS BASED ON NEAR-OPTIMAL SOLUTIONS

			Average error rate			Average CPU time(s)		
n	b	λ	IGNP	HAMC	ONP	IGNP	HAMC	ONP
30	0.05	0.2	0.009	0.049	0.091	1.755	0.297	0.319
		1.0	0.000	0.092	0.674	1.872	0.299	0.323
		3.0	0.000	0.000	0.209	1.877	0.297	0.331
	0.10	0.2	0.092	0.040	0.092	1.806	0.305	0.328
		1.0	0.000	0.092	0.277	1.862	0.297	0.338
		3.0	0.003	0.000	0.662	1.881	0.286	0.323
		Average	0.017	0.045	0.334	1.842	0.296	0.327
40	0.05	0.2	0.042	0.016	0.112	5.325	0.686	0.883
		1.0	0.000	0.117	0.301	5.741	0.684	0.872
		3.0	0.000	0.001	0.461	5.702	0.661	0.845
	0.10	0.2	0.035	0.144	0.379	5.381	0.694	0.891
		1.0	0.048	0.063	0.697	5.627	0.664	0.862
		3.0	0.002	0.000	0.401	5.661	0.642	0.878
		Average	0.021	0.056	0.391	5.572	0.671	0.871
50	0.05	0.2	0.015	0.114	0.199	12.272	1.286	1.767
		1.0	0.004	0.096	0.684	13.720	1.331	1.941
		3.0	0.000	0.000	0.596	13.597	1.281	1.966
	0.10	0.2	0.011	0.149	0.324	12.806	1.313	1.900
		1.0	0.087	0.079	0.192	13.822	1.294	2.059
		3.0	0.000	0.001	0.421	14.850	1.364	2.191
		Average	0.019	0.073	0.402	13.511	1.311	1.970

The average error rate of algorithms with respect to the job size n is shown in Figure 10. It can be seen that: the average error rate of IGNP is stable, it is hardly affected by the job size n; that of ONP increases with the job size n increase; that of HAMC is between of them.

(2) The performance of the IGNP, HAMC and ONP for solving the medium-larger scale problem is tested.

The control variable λ 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 $(n = 30, 40, 50)$ are adopted. The average error rate of IGNP, HAMC and ONP, respectively, relative to the optimal solution, are provided in Table 2. The error rate is $var = (H - H^*)/H^* \times 100\%$, where H is a solution from IGNP, HAMC or ONP, and H^* is a best solution among IGNP, HAMC or ONP. The average CPU time of the IGNP, HAMC and ONP are recorded.

From Table 2, the performance of IGNP is far better than that of HAMC and ONP. ONP obtains solutions which become inferior over the job size increase. The performance of HAMC is little lower than that of IGNP, but far prior to that of ONP.

In summary, the B&B can be used for obtaining the optimal solution when the job size is equal to or smaller than 15. It is a good choice for solving medium-small scale problem. However, for solving the large scale problem, IGNP and HAMC are good methods.

V. CONCLUSIONS

In this paper, a novel single machine scheduling problem with deterioration depending on piece-wise function is presented. Firstly, a new piece-wise 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 IGNP method and HAMC are proposed. The results of the numerical examples with a job size smaller than 15 jobs show that, the B&B algorithm can obtain the optimal solutions in a reasonable time. And HAMC can also obtain good near-optimal solutions. The IGNP is prior to HAMC since it can obtain the average error percentage of near-optimal solutions less than 0.036 within 0.2s. According to the results of the analysis, it shows the efficiency of IGNP. Therefore, they can be used for solving large size problems. In the future, we focus on the following aspects: (1) since some uncertainties exist in the real scheduling problem, risk should be integrated into it; (2) in order to satisfy decision makers demand, multi-objective will be considered; (3) moreover, rescheduling problems will be also discussed when the initial scheduling is interrupted.

REFERENCES

[1] J. Gupta and S. Gupta, "Single facility scheduling with nonlinear processing times," *Computers and Industrial Engineering*, vol. 14, no. 4, pp. 387–393, 1988.

[2] S. Browne and U. Yechiali, "Scheduling deteriorating jobs on a single processor," *Operations Research*, vol. 38, no. 3, pp. 495–498, 1990.

[3] W. L. T.C.E. Cheng and C. Wu, "Single-machine scheduling with deteriorating functions for job processing times," *Applied Mathematical Modelling*, vol. 34, no. 12, pp. 4171–4178, 2010.

[4] P. Lai and W. Lee, "Single-machine scheduling with a nonlinear deterioration function," *Information Processing Letters*, vol. 110, no. 11, pp. 455–459, 2010.

[5] C. W. P. Shen and X. Huang, "Single-machine scheduling problems with an actual time-dependent deterioration," *Applied Mathematical Modelling*, vol. 37, no. 7, pp. 5555–5562, 2013.

[6] G. P. A.J. Ruiz-Torres and E. Prez, "Parallel machine scheduling to minimize the makespan with sequence dependent deteriorating effects," *Computers & Operations Research*, vol. 40, no. 8, pp. 2051–2061, 2013.

[7] 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*, vol. 225, no. 3, pp. 547–551, 2013.
[8] Y. H. D. Wang and P. Ji, "Single-machine group scheduling with deteriorating jobs and allotted resource," *Optimization Letters*, vol. 8, no. 2, pp. 591–605, 2014.
[9] T. C. E. C. Y. Yin, W-H Wu and C.-C. Wu, "Sing-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] J. L. R.L. Graham, E.L. Lawler and A. RinnooyKan, "Optimization and approximation in the deterministic sequencing and scheduling: a survey." *Annals of Discrete Mathematics*, no. 5, pp. 287–326, 1979.
[11] T. Cheng and Q. Ding, "The complexity of scheduling starting time dependent tasks with release times," *Information Processing Letters*, vol. 65, no. 2, pp. 75–79, 1998.
[12] T. Helmy and Z. Rasheed, "Independent job scheduling by fuzzy c-mean clustering and an ant optimization algorithm in a computation grid," *IAENG International Journal of Computer Science*, vol. 37, no. 2, pp. 136–145, 2010.
[13] R. Kashyap and D. P. Vidyarthi, "Dual objective security driven scheduling model for computational grid using ga," *IAENG International Journal of Computer Science*, vol. 38, no. 1, pp. 71–79, 2012.
[14] P. P. M. Gualtieri and F. Rossi, "Heuristic algorithms for scheduling jobs on identical parallel machines via measures of spread," *IAENG International Journal of Applied Mathematics*, vol. 39, no. 2, pp. 100–107, 2009.
[15] L. Shi and S. Olafsson, "Nested partitions method for global optimization," *Operations Research*, vol. 48, no. 3, pp. 390–407, 2000.
[16] S. Olafsson and J. Yang, "Intelligent partitioning for feature selection," *INFORMS Journal on Computing*, vol. 17, no. 3, pp. 339–355, 2005.
[17] Y. P. L. Pi and L. Shi, "Hybrid nested partitions and mathematical programming approach and its applications," *IEEE Transactions on Automation Science & Engineering*, vol. 5, no. 4, pp. 573–586, 2008.
[18] S. O. L. Shi and Q. Chen, "An optimization framework for product design," *Management Science*, vol. 47, no. 12, pp. 1681–1692, 2001.
[19] 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*, vol. 37, no. 2, pp. 247–255, 2010.
[20] C. B. Chu, "A branch-and-bound algorithm to minimize total flow time with unequal release dates," *Naval Research Logistics*, vol. 39, no. 6, pp. 859–875, 1992.