Single machine scheduling with controllable processing times to minimize total tardiness and earliness

Vahid Kayvanfar, Iraj Mahdavi, GH.M. Komaki

Computers & Industrial Engineering(2013), Vol. 65, No. 1, p166-175

2016. 08. 23
Jeong Hoon Shin
Contents

Introduction

Related literature
- Early/tardy scheduling problem with controllable processing times
- The proposed net benefit compression/net benefit expansion
- The initial sequence on single machine using heuristic techniques

Computational Results

Conclusions
Introduction
Introduction

- In JIT environment
  - jobs which are completed before their due dates → earliness penalties
    : deterioration of perishable goods, opportunity costs, holding costs for finished goods
    → can represent manufacturer concerns
  - jobs which are tardy → tardiness penalties
    : lost sales, backlogging cost and loss of goodwill
    → may represent the customer concerns
  - neither earliness nor tardiness is desirable
- In this paper (assume that)
  - the normal processing time of jobs can be reduced or increased to a predefined limit.
  - no machine idle time is allowed.
- Applicable examples of controllable processing time
  - chemical industries: processing time of a job is reduced using catalyzer or increase by an inhibitor
  - CNC machining operations in manufacturing environments: control the processing time of a job by setting the cutting speed and/or the feed rate on the machine
Related literature
## Related literature

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>with common due date</td>
<td>Abdul-Razaq &amp; Potts(1988); Ow &amp; Morton(1989); Sundararaghavan &amp; Ahmed(1984)</td>
</tr>
<tr>
<td></td>
<td>Review</td>
<td>Hoogeveen(2005)</td>
</tr>
<tr>
<td></td>
<td>with exact algorithms or heuristic methods</td>
<td>Davis and Kanet(1993); Hendel and Sourd(2006); Bülbül, Kaminsky, and Yano(2007); Sourd and Kedad-Sidhoum(2003, 2008); Alvarez-Valdes, Crespo, Tamarit, and Villa(2010)</td>
</tr>
<tr>
<td>CPSP</td>
<td>General</td>
<td>Vickson(1980); Panwalkar and Rajagopalan(1992); Zdrzalka(1991); Chen, Lu, and Tang(1997); Biskup and Cheng (1999); Kayan and Akturk(2005); Atan and Akturk(2008);</td>
</tr>
<tr>
<td></td>
<td>Survey</td>
<td>Nowicki and Zdrzalka(1990); Alidaee and Ahmadian(1993); Shabtay and Steiner(2007);</td>
</tr>
</tbody>
</table>

**Contribution of this paper**

- no outstanding research in context of early/tardy scheduling problem under key assumptions of controllable processing time, no inserted idle time and no allowable preemption job is found in which a deterministic heuristic for determining a set of compression/expansion job processing times has been proposed.
Early/tardy scheduling problem with controllable processing times
Notations

- $J_j$: job $j$ ($j = 1, 2, 3, \ldots, n$)
- $J[i]$: job placed in the $i$th position in a sequence
- $p_j$: normal processing time of job $j$
- $p'_j$: crash (minimum allowable) processing time of job $j$
- $p''_j$: expansion (maximum allowable) processing time of job $j$
- $c_j$: unit cost of compression of job $j$
- $c'_j$: unit cost of expansion of job $j$
- $x_j$: amount of compression of job $j$, $0 \leq x_j \leq m_j$
- $x'_j$: amount of expansion of job $j$, $0 \leq x'_j \leq m'_j$
- $m_j$: maximum amount of compression of job $j$, $m_j = p_j - p'_j$
- $m'_j$: maximum amount of expansion of job $j$, $m'_j = p''_j - p_j$
- $t_j$: start time of job $j$
- $d_j$: due date of job $j$
- $C_j$: completion time of job $j$
- $T_j$: tardiness of job $j$, $T_j = \max \{0, C_j - d_j\}$
- $E_j$: earliness of job $j$, $E_j = \max \{0, d_j - C_j\}$
- $n$: number of jobs
- $N$: number of iterations
- $p^*_j$: actual processing time of job $j$ (reduces status)
- $p''_j$: actual processing time of job $j$ (increased status)
- $\pi$: a processing sequence of jobs
- $M$: an arbitrary big positive number
- $C(\pi)$: total cost of related sequence $\pi$
- $\gamma_j$: earliness penalties for job $j$ in sequence $\pi$
- $\beta_j$: tardiness penalties for job $j$ in sequence $\pi$

- a single machine and $n$ jobs are available at time zero simultaneously
- each job can only be compressed or expanded or none of them
- if a job is processed at its normal processing times, it undergoes no additional costs.

$$\lambda_j = \begin{cases} 1 & \text{if job } j \text{ is compressed} \\ 0 & \text{W.} \end{cases} \quad j = 1, 2, \ldots, n$$
mathematical model

\[
\min C(\pi) = \min \left( \sum_{j=1}^{n} \left( \alpha_j E_j + \beta_j T_j + c_j x_j + c'_j x'_j \right) \right)
\]  

Subject to

1. \( M y_{jk} + t_j - t_k + x_k - x'_k \geq p_k; \quad j = 1, 2, \ldots, n; \quad j \leq k \)  
2. \( M(1 - y_{jk}) + t_k - t_j + x_j - x'_j \geq p_j; \quad j = 1, 2, \ldots, n; \quad j \leq k \)  
3. \( t_j + p_j - x_j + x'_j - d_j \leq T_j; \quad j = 1, 2, \ldots, n \)  
4. \( d_j - t_j - p_j + x_j - x'_j \leq E_j; \quad j = 1, 2, \ldots, n \)  
5. \( p_j - p'_j \geq x_j; \quad j = 1, 2, \ldots, n \)  
6. \( p''_j - p_j \geq x'_j; \quad j = 1, 2, \ldots, n \)  
7. \( M \lambda_j \geq x_j; \quad j = 1, 2, \ldots, n \)  
8. \( M(1 - \lambda_j) \geq x'_j; \quad j = 1, 2, \ldots, n \)  
9. \( T_j \geq 0, \quad E_j \geq 0, \quad x_j \geq 0, \quad x'_j \geq 0, \quad t_j \geq 0, \quad \lambda_j = 0 \text{ or } 1 \)  
10. \( x_j, x'_j \text{ : int}; \quad j = 1, 2, \ldots, n \)  
11. \( y_{jk} = \begin{cases} 
1; & \text{if job } j \text{ proceeds job } k \text{ in process sequence}; \quad j, k = 1, 2, \ldots, n; \quad j < k \\
0; & \text{if } O.W.
\end{cases} \)
The proposed net benefit compression /net benefit expansion
NBC & NBE

(1) Objective function improvement by one unit decrease or increase in the PT of job \( i \) – (cost incurred by by one unit decrease or increase in the PT of job \( i \)).

\[
\begin{align*}
\text{NBC}_{[i]} &= \sum_{j \in A} \beta_j - c_{[i]} - \sum_{j \in A'} \alpha_j \quad ; A = \{j : j > i \& T_j > 0\}; A' = \{j : j > i \& j \notin A\} \quad (12) \\
\text{NBE}_{[i]} &= \sum_{j \in B} \alpha_j - c'_{[i]} - \sum_{j \in B'} \beta_j \quad ; B = \{j : j > i \& E_j > 0\}; B' = \{j : j > i \& j \notin B\} \quad (13)
\end{align*}
\]

(2) Two variables for determining whether a given job can be compress or expand.

\[
Q_{[i]} = \begin{cases} 
1; & \text{if } p''_{[i]} - p'_{[i]} > 0 \\
0; & \text{if } p''_{[i]} - p'_{[i]} = 0 
\end{cases}
\]

\( Q_{[i]} \) represents the decision to compress or maintain the PT, where \( p''_{[i]} \) is the actual PT in the reduces status, and \( p'_{[i]} \) is the actual PT in the increase status. Min PT refers to the condition for compression.

\[
Q'_{[i]} = \begin{cases} 
1; & \text{if } p''_{[i]} - p^{**}_{[i]} > 0 \\
0; & \text{if } p''_{[i]} - p^{**}_{[i]} = 0 
\end{cases}
\]

\( Q'_{[i]} \) represents the decision to expand the PT, with \( p''_{[i]} \) being the actual PT in the increase status, and \( p^{**}_{[i]} \) being the actual PT in the reduce status. Max PT refers to the condition for expansion.
NBC-NBE algorithm

(1) Set \( x_i = x_i' = 0, p_i^{**} = p_i^* = p_i \), and calculate \( T_{[i]}, E_{[i]} \), \( \forall i = 1, 2, \ldots, n \) for a given job sequence \( \pi \).

(2) Calculate \( NBC_{[i]}, NBE_{[i]} \); \( \forall i = 1, 2, \ldots, n \). If \( Q_{[i]} = 0 \), set \( NBC_{[i]} = 0 \) and if \( Q_{[i]}' = 0 \), set \( NBE_{[i]} = 0 \). If there is no job with \( NBC_{[i]} > 0 \) and \( NBE_{[i]} > 0 \), go to step 4.

(3) Select \( \max\{NBE_{[i]}, NBC_{[i]}\} \); \( \forall i = 1, 2, \ldots, n \) and call the selected job \( J_{[k]} \). If two values of NBCs or NBEs are the same, there may be occur three states:

(3.1) NBC (if both selected jobs have tardiness)

i. For both selected jobs, compute 
\[
V = \sum_{j \in A} \beta_j - \sum_{j \in A} \gamma_j.
\]

ii. Select the job with \( \max \{V\} \) and decrease its processing time.

(3.2) NBE (if both selected jobs have earliness)

i. For both selected jobs, compute 
\[
W = \sum_{j \in B} \gamma_j - \sum_{j \in B} \beta_j.
\]

ii. Select the job with \( \max \{W\} \) and increase its processing time.

(3.3) NBC-NBE (if one of the selected job has tardiness and another one has earliness)

i. Calculate \( V \) for the tardy and \( W \) for the early job with respect to the same job.

ii. Select the job with the maximum value and then do the related action (compress for selecting the tardy job or expand for the early one). If these values are equal for both jobs, select the job which has more latter tardy jobs in sequence (with respect to the same job) and do the related action, i.e., compress for selecting the tardy job or expand for the early one (for the sake of priority of decreasing tardiness versus earliness).

Now, update the real processing time of \( J_{[k]} \) by \( p_{[k]} = p_{[k]}^{**} - 1 \) (compressing status) or \( p_{[k]} = p_{[k]}^{**} + 1 \) (expansion status), the objective function value, \( Z \), and \( T_{[l]}, E_{[l]} \) for all \( l > k \). Return to step 2.

(4) Calculate \( x_{[l]} \) and \( x_{[l]}' \) by
\[
\begin{cases} 
  x_{[l]} = p_{[l]}^{**} - p_{[l]}^* \\
  x_{[l]}' = p_{[l]} = p_{[l]}' 
\end{cases} \quad \forall i = 1, 2, \ldots, n.
\]
Stop.

- Steps 2 and 3 repeat until no NBC or NBE is positive.
- Deterministic algorithm
The initial sequence on single machine using heuristic techniques
Greedy randomized dispatching rule (1)

- greedy randomized dispatching rule with improvement steps. (Valente and Moreira (2009))

1. Generate initial ETP sequence and calculate its objective function value \(\text{ofv}\). Set \(S_{\text{best}}\) and \(\text{ofv}_{\text{best}}\) to the ETP sequence and its objective function, respectively.
2. Set \(\text{iter\_no\_improve} = 0\).
3. While \(\text{iter\_no\_improve} < \text{max\_iter\_no\_improve}\):
   - 3.1. Apply Proc. 1 and set the obtained sequence and objective function \(S\) and \(\text{ofv}\); respectively.
   - 3.2. If \(\text{ofv} < \text{ofv}_{\text{best}}\), set \(S_{\text{best}} = S\), \(\text{ofv}_{\text{best}} = \text{ofv}\) and \(\text{iter\_no\_improve} = 0\). Otherwise, set \(\text{iter\_no\_improve} = \text{iter\_no\_improve} + 1\).
4. Apply 3SW improvement procedure to the best schedule found \(S_{\text{best}}\). Set \(S_{\text{best}}\) and \(\text{ofv}_{\text{best}}\) to the sequence and \(\text{ofv}\), respectively, of the schedule obtained after the application of the improvement procedure.
5. Return \(S_{\text{best}}\) and its objective function \(\text{ofv}_{\text{best}}\).

- Stop criterion: use max consecutive iteration of no improvement
- 3SW improvement: the best configuration from all possible permutations of three consecutive jobs is selected. The procedure is applied repeatedly until no improvement is possible.

ETP rule: non increasing order of \(I_j(t)\). (Valente and Alves (2008))

\[
I_j(t) = \begin{cases} 
(w_j/p_j)[\bar{p} + 2(t + p_j - d_j)] & \text{if } s_j \leq 0, \\
T_j^0 - s_j(T_j^0 - E_j^{kp})/k\bar{p} & \text{if } 0 < s_j < k\bar{p}, \\
(h_j/p_j)[\bar{p} - 2(d_j - t - p_j)] & \text{if } s_j \geq k\bar{p}.
\end{cases}
\]

\[
T_j^0 = \text{the priority value of job } j \text{ when } s_j = 0 \\
T_j^0 = (w_j/p_j)\bar{p} \\
E_j^{kp} = \text{the priority value of job } j \text{ when } s_j = k\bar{p} \\
E_j^{kp} = (h_j/p_j) [\bar{p} - 2k\bar{p}]
\]
Greedy randomized dispatching rule (2)

Procedure 1:
1. Set \( S = \emptyset \) and \( U = \{1, 2, \ldots, n\} \).
2. While \( U \neq \emptyset \):
   2.1. Calculate the priority value \( \eta_j \) for all jobs \( j \in U \).
   2.2. Create candidate list (CL) of unscheduled jobs that will be considered to be scheduled in the next position.
   2.3. Calculate the score \( sc_j \) for all jobs in CL.
   2.4. Calculate the biased score \( bsc_j \) for all jobs in CL.
   2.5. Calculate the probability \( prob_j \) of selecting each job in CL:
   \[
   Prob_j = bsc_j / \sum_{j \in CL} bsc_j
   \]
   2.6. Randomly select the next job to be scheduled from the jobs in CL according to the probabilities \( prob_j \).
   2.7. Add the selected job to set \( S \) and remove it from set \( U \).
3. Return the \( S \).

2.2 the candidate list can be all unscheduled jobs (ALL)
2.3 value-biased (VB) stochastic sampling approach is used to calculate the job score(\( sc_j \)).
In this paper: \( sc_j = \eta_j \)
2.4 the exponential bias function: \( bsc_j = \exp - base^{sc_j} \)
In this paper: \( \exp - base = 1 + 0.1n^{-0.33} \)
(Valente and Moreira (2009))

From Ow and Morton (1989)

\[
\eta(j) = \begin{cases} 
\beta_j/p_j; & s_j < 0 \\
-\alpha_j/p_j + (\alpha_j/p_j + \beta_j/p_j) \exp(-s_j/k\bar{p}); & 0 \leq s_j \leq k\bar{p} \\
-\alpha_j/p_j; & s_j > k\bar{p}
\end{cases}
\]

\( s_j = d_j - t - p_j \)
early/tardy dispatch priority rule

- early/tardy dispatch priority rule.
  - In Ow and Morton (1989) calculate the priority of each job by Eq. (16).

- Improved early/tardy dispatch priority rule (from Li (1997)).
  - J: current partial schedule (m jobs); S: the set of unscheduled jobs (k jobs)
  - early/tardy dispatch priority rule is used to schedule the jobs in S following J
  - for each job in S, a corresponding schedule of all jobs, will be obtained and the total early and tardy cost of this schedule is assigned to job j (first assigned job following J) as its cost value.
  - Select job i in S with the smallest cost value to schedule next.
**early/tardy dispatch priority rule**


1. If number of jobs is less than 500, use the new dispatch priority rule (the improved one) to obtain an initial schedule as the seed; otherwise, use the early/tardy dispatch priority rule to get this goal. Let $OP_d$ with $d = 0$ be the initial operator. Set the stopping criterion. ($OP_d :$ pairwise interchange of jobs i and j with d jobs between them)

2. Start with first or last job in the seed, apply the current operator to produce a neighborhood of the seed, and calculate the total early and tardy cost for each schedule in the neighborhood. If one of the schedules in the neighborhood has a lower total early and tardy cost than the seed, go to step 3. Otherwise, if stopping criterion is met, stop; if not, update the operator by $d = d + 1$ (if $d < k$) or $d = 0$ if $d = k$ ($k$ is an empirical integer parameter) and go to step 2 (try to move out of local optima).

3. Select one of the schedules in the neighborhood which has the lower total early and tardy cost than the seed. Put this schedule as the new seed and go to step 2.
Hybrid NEH–E/T dispatch priority rule

- This algorithm is combination of Li(1997)’s algorithm and NEH algorithm → HNET.
  - Li’s algorithm is firstly applied to obtain initial sequence
  - NEH algorithm is then employed (for improvement)
- The steps of HNET.
  1. Apply the dispatch rule proposed by Li (1997) and set the obtained sequence $S_{GL}$.
  2. Set $k=2$. Pick the two first jobs from $S_{GL}$ and find the best possible sequence for these two jobs.
  3. Set $k=k+1$ and pick the k-th job in the $S_{GL}$. Insert it into k possible positions of the best partial sequence found so far. Pick the best k-job partial sequence among these k sequences based on "minimizing total tardiness and earliness" criterion. Now, insert each job (except the k-th job of $S_{GL}$) into k-1 possible positions and choose the best partial sequence among generated sequences.
  4. If $k = n$, stop; otherwise go to step 2.
Example of NBC–NBE algorithm

**Initial sequence** $\pi = J_4 - J_3 - J_1 - J_2 - J_5$.

<table>
<thead>
<tr>
<th>Job #</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>due</td>
<td>6</td>
<td>7</td>
<td>12</td>
<td>15</td>
<td>20</td>
</tr>
</tbody>
</table>

**Table 1**
Input data for a five job problem.

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>3</th>
<th>1</th>
<th>2</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_j$</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>$p_j'$</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>$p_j''$</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>$d_j$</td>
<td>6</td>
<td>7</td>
<td>12</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>$c_j$</td>
<td>0.2</td>
<td>0.1</td>
<td>0.05</td>
<td>0.1</td>
<td>0.15</td>
</tr>
<tr>
<td>$c_j'$</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.05</td>
<td>0.2</td>
</tr>
<tr>
<td>$x_j$</td>
<td>2.5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.25</td>
</tr>
<tr>
<td>$\beta_j$</td>
<td>1</td>
<td>1.5</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
</tr>
</tbody>
</table>

$NBC_{[i]} = \sum_{j \in A} \beta_j - c_{[i]} - \sum_{j \in A'} x_j$

$NBE_{[i]} = \sum_{j \in B} x_j - c_{[i]}' - \sum_{j \in B'} \beta_j$
Computational Results
Test instances & small size problem results

Due date: \( P = \sum_{j=1}^{n} p_j \), \( \rho \) is parameter
\( \rho = 0.2, 0.4, 0.6, 0.8, 1.0 \)

# of instances per # of jobs: 5(# of \( \rho \)) \times 10

Total instances: 6(# of jobs) \times 5 \times 10 = 300

Small size problem: \( n = 7 \)

medium-to-large size problems:
\( n = 15, 30, 50, 100, 200 \)

percentage relative error (PRE)
\[
PRE = \frac{\text{Alg}\_sol - O}{O} \times 100
\]

- \text{Alg}\_sol: the objective value obtained by selected heuristic.
- \( O \): the optimum value obtained by lingo.

MCPU time: mean of CPU Time for all \( \rho \) in each size of problems

<table>
<thead>
<tr>
<th>( \rho )</th>
<th>Lingo</th>
<th>GNBC–NBE</th>
<th>DNBC–NBE</th>
<th>HNBC–NBE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCPU time</td>
<td>MCPU time</td>
<td>( PRE_{avg} )</td>
<td>MCPU time</td>
</tr>
<tr>
<td>0.2</td>
<td>1520.36</td>
<td>0.028</td>
<td>16.61</td>
<td>0.022</td>
</tr>
<tr>
<td>0.4</td>
<td>697.06</td>
<td>0.025</td>
<td>19.86</td>
<td>0.022</td>
</tr>
<tr>
<td>0.6</td>
<td>507.96</td>
<td>0.029</td>
<td>36.11</td>
<td>0.030</td>
</tr>
<tr>
<td>0.8</td>
<td>1184.02</td>
<td>0.024</td>
<td>7.15</td>
<td>0.023</td>
</tr>
<tr>
<td>1.0</td>
<td>362.08</td>
<td>0.024</td>
<td>14.85</td>
<td>0.023</td>
</tr>
<tr>
<td>Mean</td>
<td>854.30</td>
<td>0.026</td>
<td>18.92</td>
<td>0.024</td>
</tr>
</tbody>
</table>

Table 2 Data set distribution.

<table>
<thead>
<tr>
<th>Input variables</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of jobs (( n ))</td>
<td>7, 15, 30, 50, 100 and 200</td>
</tr>
<tr>
<td>normal processing time (( p_i ))</td>
<td>( \sim DU(3, 25) )</td>
</tr>
<tr>
<td>crash processing time (( p'_i ))</td>
<td>( \sim DU(0.5 \times p_i, p_i) )</td>
</tr>
<tr>
<td>expansion processing time (( p''_i ))</td>
<td>( \sim DU(p_i, 1.5 \times p_i) )</td>
</tr>
<tr>
<td>Due dates (( d_j ))</td>
<td>( \sim DU[\max(0, P(1 - 3 \rho/2)), P(1 + \rho/2)] )</td>
</tr>
<tr>
<td>unit cost of compression (( c_j ))</td>
<td>( \sim U(0.1, 2.5) )</td>
</tr>
<tr>
<td>unit cost of expansion (( c'_j ))</td>
<td>( \sim U(0.1, 2.5) )</td>
</tr>
<tr>
<td>Earliness penalty (( \alpha_j ))</td>
<td>( \sim U(0.5, 2.5) )</td>
</tr>
<tr>
<td>Tardiness penalty (( \beta_j ))</td>
<td>( \sim U(0.5, 4.5) )</td>
</tr>
</tbody>
</table>
medium-to-large size problem results

Table 4  Computational result for medium-to-large size problems.

<table>
<thead>
<tr>
<th>N</th>
<th>( \rho )</th>
<th>SA</th>
<th>GA</th>
<th>GNBC-NBE</th>
<th>DNBC-NBE</th>
<th>HNBC-NBE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MCPU time</td>
<td>( RPD_{avg} )</td>
<td>MCPU time</td>
<td>( RPD_{avg} )</td>
<td>MCPU time</td>
</tr>
<tr>
<td>15</td>
<td>0.2</td>
<td>0.050</td>
<td>29.68</td>
<td>68.71</td>
<td>7.43</td>
<td>0.030</td>
</tr>
<tr>
<td>Mean of all</td>
<td></td>
<td>0.608</td>
<td>47.73</td>
<td>909.716</td>
<td>12.05</td>
<td>0.436</td>
</tr>
</tbody>
</table>

HNBC-NBE outperforms all algorithms in average with 7.489% RPD.

Relative percentage deviation (RPD):

\[
RPD = \frac{Alg_{sol} - Min_{sol}}{Min_{sol}} \times 100
\]

- \( Alg_{sol} \): the objective value obtained by selected heuristic.
- \( Min_{sol} \): the best solution obtained for each instance.

Fluctuations of RPD in HNBC–NBE is negligible (less than 5%), except 200 jobs.
Conclusions

- In this paper, study the single machine earliness-tardiness problem (SMETP) regarding controllable processing times.
- A NBC–NBE heuristic is presented.
- Employ three heuristics for initial sequence of jobs.
- Experiments are performed three heuristic combination compared with optimal solution (small size problem (n=7)) and SA & GA (medium to large problem).
- Experimental results shows that proposed HNBC_NBE heuristic outperforms all other algorithms in terms of average RPD.
감사합니다