Solving Single Machine Scheduling Problem With Common Due Date
Nordin Haji Mohamad and Fatimah Said

Abstract
The paper addresses an n-job single machine scheduling problem with common due date to minimize the sum of total inventory and penalty costs. Earliness and tardiness are considered harmful to profitability. Earliness causes inventory carrying costs and possible loss of product quality, while tardiness causes loss of customer goodwill and damage reputation as well as delay of payment. Thus the scheduling problem of minimizing the total sum of earliness and tardiness with a common due date on a single machine is important and a competitive task in the delivery of goods in production plants, and it is known to be NP-hard. A simple, easy to understand/implement heuristic algorithm which can be performed manually for small problems and computationally feasible for large problems is presented and illustrated with numerical example.

Key words: Jobs scheduling problem, due date, earliness, tardiness

INTRODUCTION
In today’s complex industrial world, many business problems that need to be solved or optimized are scheduling problems. A manufacturing firm producing multiple products, each requiring many different processes and machine facilities for completion, must find a way to successfully manage resources in the most efficient way possible. The decision maker is faced with a problem of designing a production or job schedule that promotes on time delivery and minimizes objectives such as the flow time or completion time of a product. Out of these interests emerged an area of study known as the scheduling problem.

A frequently occurring scheduling problem is one of processing a given number of jobs or tasks on a specified number of machines or facilities. This class of problem also referred to by many as dispatching or sequencing is categorized as NP-hard. The desire to process the jobs in a specific order to achieve some objective function is what creates a problem that remains largely unsolved. The actual situations that give rise to scheduling problems are wide and varied. This includes, for example, single machine scheduling problem, multiple machine scheduling problem and manpower scheduling problem.

A general n jobs m machines scheduling problem can be stated as follows. Given n jobs to be processed on m machines in the same technological order, the processing time of job i on machine j being $t_{ij}$ ($i = 1, 2, ..., n; j = 1, 2, ..., m$), it is desired to find the order (schedule or sequence) in which these n jobs should be processed on each of the m machines so as to optimize (minimize or maximize) a well defined measure of some objectives (such as production cost, number of late jobs, etc). This problem, in general, gives $(n!)^m$ possible schedules. Even for problems as small as $n = m = 5$, the number of possible schedules is so large that a direct enumeration is economically impossible. (For $n = 5$, $m = 3$, we have 1,728,000 possible schedules, whereas for $n = m = 5$, we have $2.48882 \times 10^{10}$ possible schedules). However, for a simplified version where it is assumed that the order (or sequence) in which these jobs are processed on each machine is the same, the number of feasible schedules reduces to $n!$.

Single machine scheduling problems bear complex computations and the analysis of such problems is important for a better understanding of the problem. Among single machine problems, those related to earliness and tardiness is more important. Completing jobs or tasks earlier than their due dates should be discouraged as much as completing jobs or tasks later than their due dates. In real world, since a customer expects to receive the product on a specific date, scheduling based on the due date is also an important task in the production planning. Earliness leads to inventory and maintenance carrying costs while tardiness leads to customer’s dissatisfaction and losing goodwill and reputation.

1 Institute of Mathematical Sciences, Faculty of Science, University of Malaya, Lembah Pantai, 50603 Kuala Lumpur, Malaysia. Telephone: 603-7967 4034, Fax: 603-7967 4143. Email: nordinhm@um.edu.my
2 Faculty of Economics and Administration, University of Malaya, 50603 Kuala Lumpur, Malaysia. fatimahs@um.edu.my

©Society for Business and Management Dynamics
LITERATURE REVIEW

Job scheduling or sequencing has a wide variety of applications, from designing the product flow and order in a manufacturing facility to modeling queues in service industries. It is indeed a useful subject that is still being actively researched. Most researchers focused on special cases in which situational restrictions are imposed. Moore (1968) designed an algorithm that sequences the jobs for the single-machine problem to minimize the number of tardiness jobs, whereas Gupta (1969) proposed a general algorithm for the $n \times m$ flowshop scheduling problem. Brucker et al. (1999) showed that complex scheduling problems like general shop problem can be reduced to single-machine problem with positive and negative time-lags between jobs, solvable by a branch and bound algorithm. Scheduling multi-machine problems considering both earliness and tardiness penalties was surveyed by Lauff and Werner (2004) which incorporated the just-in-time (JIT) production philosophy. A comprehensive review of some earliness and tardiness models can be found in Baker and Scudder (1990) where seven different objective functions associated with the minimization of variations of job completion times from their respective due dates were identified, including cases of nonlinear penalties. Tavakkoli-Moghaddam et al. (2005) consider the common due date problem with the objective of minimizing the sum of maximum earliness and tardiness costs using an idle insert algorithm and illustrated the efficiency of the proposed algorithm to 1020 problems with different job sizes. A linear programming approach to solve a fuzzy single machine scheduling problem is proposed by Kamalabadi et al. (2007). This approach is applicable to just-in-time systems, in which many firms face the need to complete jobs as close as possible to their due dates. A recent study by Gupta (2011) proposes a heuristic algorithm for small system with distinct due dates under fuzzy environment.

In addition, the integer programming method for solving problems with small size of jobs was raised by Biskup and Feldmann (2001). Ronconi and Kawamura (2010) proposed a branch-and-bound algorithm for solving single machine earliness and tardiness scheduling problem. They introduced lower bounds and pruning that exploit properties of the problem. Feldmann and Biskup (2003) studied single machine scheduling problems using three meta-heuristics approaches (evolutionary search, simulated annealing and threshold accepting). The application of these meta-heuristics was demonstrated by solving 140 benchmark problems with up to 1000 jobs. Several meta-heuristic algorithms for solving single machine scheduling problems were analyzed by Abtahi and Taghavifard (2008).

In this paper, we consider a simplified version of scheduling $n$ jobs with common due date on a single machine with the objective of minimizing the sum of total earliness and tardiness penalties. Common due date problems are relevant in many real-life situations: for instance, if a customer orders a bundle of goods which has to be delivered at a specified time, if a firm has installed a weekly bulk delivery to the wholesaler or in an assembly environment in which the components of a product should all be ready at the same time to avoid staging delays (Yang and Hsu, 2010). Among the pioneers studying common due date problems were Kanet (1981) and Panwalker et al. (1982). All the jobs considered have a common due date. The objective was to find an optimal common due date and an optimal schedule which minimizes the total earliness, tardiness and due date costs. Since then, the problems have been studied under different environments. Cheng et al. (2004) studied a single machine due date assignment scheduling with deteriorating jobs. They provided some properties and an algorithm to solve the problem in $O(n \log n)$ time. Later, Kuo and Yang (2008) gave a concise analysis of the problem and provided a simpler algorithm for the problem. Comprehensive survey on this topic is provided by Cheng and Gupta (1989), Baker and Scudder (1990) and Gordon et al. (2002).

STATEMENT OF THE PROBLEM

One of the most important objectives in scheduling problem with due dates is to minimize the sum of the earliness and tardiness of jobs. This conforms to the JIT system (Ow and Morton, 1989). Earliness and tardiness cause penalties in increasing inventory cost and losing customers respectively. Early jobs (completed before due dates) tie up capital, increase the inventory level, take up scarce floor space, cause losses owing to deterioration, and generally indicate sub-optimal resource allocation and utilization, whereas late or tardy jobs (completed after due dates) result in penalties, such as loss of customers goodwill and damaged reputation.
There are many real life problems that resemble single machine scheduling problem. A typical example is the laundry service where orders (of different sizes) from customers arrive early morning, and due dates are determined by pick-up times, and pick-ups are made by customers. If a due date (pick-up) is missed, a special delivery service needs to be bought by the laundry operator, the cost of which is independent of the tardiness. Other examples include a single contractor scheduling multiple building/housing projects with due date completion times and production of manufacturing goods with different processing times to meet delivery’s deadlines.

The general single-machine problem with common due date can thus be formally stated as follows.

- Given $n$ jobs to be processed on a single machine, the processing time of job $i$, being $t_i$, $i = 1, 2, \ldots, n$. It is assumed that all jobs are ready for processing at time zero and have the same common due date (deadline). Also, no more than one job can be processed at any point of time. Each job $i$ requires exactly one operation and its processing time $p_i$ is known. If a job $i$ is completed before the due date, its earliness is given by $E_i = d - c_i$ where $c_i$ is the completion time of job $i$. Conversely, if a job $i$ is completed after the desired date, its tardiness is given by $T_i = c_i - d$. Each job $i$ has its own unit earliness penalty $\alpha_i$ and unit tardiness penalty $\beta_i$. The problem is to find the order (schedule) in which these $n$ jobs should be processed so as to minimize the sum of total earliness and tardiness costs.

- It is also assumed that the due date is less than the total processing times, a problem often referred to as restricted due date problem. A due date is called unrestrictive if its optimal value has to be calculated or if it’s given value does not influence the optimal schedule (Ronconi and Kawamura, 2010). If the given due date is greater than or equal to the sum of processing times of all jobs available, the problem is unrestrictive (Feldmann and Biskup, 2003). Hall and Posner (1991) showed that this scheduling problem is NP-hard even with $\alpha_i = \beta_i$. Ghosh and Wells (1994) addressed the unrestrictive case in which $\alpha_i = \beta_i = 1$ for all jobs that can be solved by a polynomial algorithm of $O(n\log n)$ complexity.

- The restrictive due date problem is NP-hard even with $\alpha_i = \beta_i = 1$ (Hall et al. 1991). Due to its complexity, many authors addressed this problem using heuristic and metaheuristic approaches (Feldmann and Biskup, 2003; Hino et al. 2005; Liao and Cheng, 2007).

The problem can be mathematically formulated as

$$\text{minimize : } F = \sum_{i=1}^{n} \{\alpha_i \max(d - c_i, 0) + \beta_i \max(c_i - d, 0)\} \quad \ldots \quad (1)$$

where $c_i$, $i = 1, 2, \ldots, n$, is the completion time of job $i$, $d$ is the common due date, and $\alpha_i$ and $\beta_i$ are the unit penalty costs associated with earliness and tardiness respectively. To illustrate the problem, we consider a simple 2-job and 3-job examples with $\alpha_i = \alpha$ and $\beta_i = \beta$, for all $i = 1, 2, \ldots, n$.

**Example 1 (case $n = 2$)**

Let $J_1$ and $J_2$ be two jobs with processing times 3 and 5 days respectively. Further, let the common due date, $d = 6$. In other words, both jobs must be delivered on day 6. The problem can be represented as in Table 1.

<table>
<thead>
<tr>
<th>$J_i$</th>
<th>$J_1$</th>
<th>$J_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

There exist two schedules: $S_1 = \{J_1, J_2\}$ and $S_2 = \{J_2, J_1\}$. Schedule $S_1$ implies job $J_1$ is processed first, followed by job $J_2$, whereas schedule $S_2$ implies the opposite, job $J_2$ is processed first, followed by job $J_1$. 

©Society for Business and Management Dynamics
For schedule $S_1 = \{J_1, J_2\}$, we have

<table>
<thead>
<tr>
<th>$J_i$</th>
<th>$J_1$</th>
<th>$J_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>$</td>
<td>c_i - d</td>
<td>= 3 - 6 = 3,</td>
</tr>
</tbody>
</table>

Therefore, total earliness and tardiness costs, $F_1 = 3\alpha + 2\beta$.

For schedule $S_2 = \{J_2, J_1\}$, we have

<table>
<thead>
<tr>
<th>$J_i$</th>
<th>$J_2$</th>
<th>$J_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>$</td>
<td>c_i - d</td>
<td>= 5 - 6 = 1,</td>
</tr>
</tbody>
</table>

Therefore, total earliness and tardiness costs, $F_2 = \alpha + 2\beta$.

Thus, $F^* = \text{minimum } (F_1, F_2) = F_1 = \alpha + 2\beta$ for all $\alpha$ and $\beta$. The optimal decision is to schedule job $J_2$ first (which is completed one day before the due date, giving an earliness cost of $\alpha$) followed by job $J_1$ (with a delay of two days and tardiness cost of $2\beta$).

Example 2 (case $n = 3$)
Let $J_1$, $J_2$ and $J_3$ be three jobs with processing times 3, 4 and 6 days respectively. Further, let the common due date, $d = 9$. In other words, all jobs must be delivered on day 9. The problem can be represented as in Table 2.

Table 2 (case $n = 3$, with due date, $d = 9$)

<table>
<thead>
<tr>
<th>$J_i$</th>
<th>$J_1$</th>
<th>$J_2$</th>
<th>$J_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

There exist six schedules: $S_1 = \{J_1, J_2, J_3\}$, $S_2 = \{J_1, J_3, J_2\}$, $S_3 = \{J_2, J_1, J_3\}$, $S_4 = \{J_2, J_3, J_1\}$, $S_5 = \{J_3, J_1, J_2\}$, $S_6 = \{J_3, J_2, J_1\}$.

Schedule $S_1$ implies job $J_1$ is processed first, followed by job $J_2$ and finally job $J_3$, whereas schedule $S_2$ implies job $J_1$ is processed first, followed by job $J_3$, and finally job $J_2$. All other schedules should be interpreted accordingly.

For schedule $S_1 = \{J_1, J_2, J_3\}$, we have

<table>
<thead>
<tr>
<th>$J_i$</th>
<th>$J_1$</th>
<th>$J_2$</th>
<th>$J_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>$</td>
<td>c_i - d</td>
<td>= 3 - 9 = 6,</td>
<td>7 - 9</td>
</tr>
</tbody>
</table>

Therefore, total earliness and tardiness costs, $F_1 = 8\alpha + 4\beta$.

For schedule $S_2 = \{J_1, J_3, J_2\}$, we have

<table>
<thead>
<tr>
<th>$J_i$</th>
<th>$J_1$</th>
<th>$J_3$</th>
<th>$J_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>3</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$</td>
<td>c_i - d</td>
<td>= 3 - 9 = 6,</td>
<td>9 - 9</td>
</tr>
</tbody>
</table>

Therefore, total earliness and tardiness costs, $F_2 = 6\alpha + 4\beta$. 

©Society for Business and Management Dynamics
For schedule $S_3 = \{J_2, J_1, J_3\}$, we have

\[
\begin{array}{cccc}
J_i & J_2 & J_1 & J_3 \\
4 & 6 & 3 & 6 \\
|c_i - d| & 4 - 9 & 7 - 9 & 13 - 9 = 4
\end{array}
\]

Therefore, total earliness and tardiness costs, $F_3 = 7\alpha + 4\beta$.

For schedule $S_4 = \{J_2, J_3, J_1\}$, we have

\[
\begin{array}{cccc}
J_i & J_2 & J_3 & J_1 \\
4 & 6 & 3 & 3 \\
|c_i - d| & 4 - 9 & 10 - 9 & 13 - 9 = 4
\end{array}
\]

Therefore, total earliness and tardiness costs, $F_4 = 5\alpha + 5\beta$.

For schedule $S_5 = \{J_3, J_1, J_2\}$, we have

\[
\begin{array}{cccc}
J_i & J_3 & J_1 & J_2 \\
6 & 3 & 4 & 4 \\
|c_i - d| & 6 - 9 & 9 - 9 & 13 - 9 = 4
\end{array}
\]

Therefore, total earliness and tardiness costs, $F_5 = 3\alpha + 4\beta$.

For schedule $S_6 = \{J_3, J_2, J_1\}$, we have

\[
\begin{array}{cccc}
J_i & J_3 & J_2 & J_1 \\
6 & 4 & 3 & 3 \\
|c_i - d| & 6 - 9 & 10 - 9 & 13 - 9 = 4
\end{array}
\]

Therefore, total earliness and tardiness costs, $F_6 = 3\alpha + 5\beta$.

Thus, $F^* = \min (F_1, F_2, F_3, F_4, F_5, F_6) = F_5 = 3\alpha + 4\beta$ for all $\alpha$ and $\beta$. The optimal decision is to schedule job $J_3$ first (which is completed three days before the due date, giving an earliness cost of $3\alpha$) followed by job $J_1$ (which is completed on time), and lastly by job $J_2$ (with a delay of four days and tardiness cost of $4\beta$).

From both examples, we observe that
- the optimal schedule appears to be independent of the numerical values of $\alpha$ and $\beta$. Thus without loss of generality and for ease of computation, we can consider a case with $\alpha = \beta = 1$
- there are $n!$ schedules for $n$ jobs. For 10 jobs, there are $10! = 3,628,800$ possible schedules which are not manually feasible. Thus an algorithm capable of reducing the number of enumerations is much desired.

**A HEURISTIC ALGORITHM**

Below, we present a heuristic algorithm for solving $n$-job single machine scheduling problem with common due date. The algorithm is simple to understand and easy to implement.

- **Problem Statement**: Given $n$ jobs to be processed on a single machine, the processing time of job $i$ being $t_i$, $i = 1, 2, ..., n$. It is assumed that all jobs are ready for processing at time zero and have the same common due date (deadline), $D$. The problem is to find the order (schedule) in which these $n$
jobs should be processed so as to minimize the sum of total earliness and tardiness costs. The common due date, $D$ is assumed to be less than the total processing time.

**Step 0:** Sort and number the $n$ jobs in non-increasing order of processing time $t_i$, $(i = 1, 2, \ldots, n)$ such that $t_1 \geq t_2 \geq \ldots \geq t_{i-1} \geq t_i \geq t_{i+1} \geq \ldots \geq t_n$. In general, we can represent the jobs in tabular form:

<table>
<thead>
<tr>
<th>$j_i$</th>
<th>$j_1$</th>
<th>$j_2$</th>
<th>$j_{i-1}$</th>
<th>$j_i$</th>
<th>$j_{i+1}$</th>
<th>$\ldots$</th>
<th>$j_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>$t_1$</td>
<td>$t_2$</td>
<td>$t_{i-1}$</td>
<td>$t_i$</td>
<td>$t_{i+1}$</td>
<td>$\ldots$</td>
<td>$t_n$</td>
</tr>
</tbody>
</table>

Compute

- total processing time, $T = \sum_{i=1}^{n} t_i$,
- $T_0 = T - D$, and $E_0 = D$.

Introduce two empty sets, $S^E_0 = \phi$ and $S^T_0 = \phi$.

**Step 1:** Consider job $j_1$ with processing time $t_1 = \max (t_i, i = 1, 2, \ldots, n)$.

Set $T_1 = T_0$, and $E_1 = E_0$.

- If $T_i < E_i$, set $S^E_i = S^E_{i-1} + \{j_i\}$ and $S^T_i = S^T_{i-1}$.
- If $T_i \geq E_i$, set $S^T_i = S^T_{i-1} + \{j_i\}$ and $S^E_i = S^E_{i-1}$.

**Step $i$:** Consider job $j_i$ with processing time $t_i$, $(1 < i \leq n)$.

- If previous job, $j_{i-1} \in S^E_{i-1}$, compute $T_i = T_{i-1}$ and $E_i = E_{i-1} - t_{i-1}$.
- If previous job, $j_{i-1} \in S^T_{i-1}$, compute $T_i = T_{i-1} - t_{i-1}$ and $E_i = E_{i-1}$.

**Assignment decision**

- If $T_i < E_i$, set $S^E_i = S^E_{i-1} + \{j_i\}$ and $S^T_i = S^T_{i-1}$.
- If $T_i \geq E_i$, set $S^T_i = S^T_{i-1} + \{j_i\}$ and $S^E_i = S^E_{i-1}$.

Iteration terminates when all jobs have been assigned to either $S^E_n$ or $S^T_n$.

**Scheduling decision**

Check (and sort if necessary) so that:

- $S^E_n = \{\text{jobs in non-increasing order of processing times}\}$,
- $S^T_n = \{\text{jobs in non-decreasing order of processing times}\}$.

The optimal schedule, $S^* = \{S^E_n, S^T_n\}$. In other words, jobs are schedule according to their sequence in $S^E_n$, followed by $S^T_n$.

Note that the procedure only involves $n$ enumerations (iterations) as compared to $n!$ possible schedules. We illustrate the above algorithm by considering a 10-job scheduling problem.

**Illustrative example**

Consider a 10-job scheduling problem given by the table below (sorted in non-increasing order of processing times), with common due date, $D = 45$.

<table>
<thead>
<tr>
<th>$j_i$</th>
<th>$j_1$</th>
<th>$j_2$</th>
<th>$j_3$</th>
<th>$j_4$</th>
<th>$j_5$</th>
<th>$j_6$</th>
<th>$j_7$</th>
<th>$j_8$</th>
<th>$j_9$</th>
<th>$j_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>20</td>
<td>18</td>
<td>15</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

**Step 0:** Compute
- total processing time, \( T = \sum_{i=1}^{10} t_i = 102 \),
- \( T_0 = T - D = 102 - 45 = 57 \), and \( E_0 = D = 45 \).

Introduce two empty sets, \( S_0^E = \phi \) and \( S_0^T = \phi \).

**Step 1:** Consider job \( J_1 \) with processing time \( t_1 = 20 \).
- Set \( T_1 = T_0 = 57 \), and \( E_1 = E_0 = 45 \).
- \( T_1 > E_1 \Rightarrow S_1^E = S_0^E + \{ J_1 \} = \{ J_1 \} \) and \( S_1^T = S_0^T = \{ \} \).

**Step 2:** Consider job \( J_2 \) with processing time \( t_2 = 18 \).
- Previous job, \( J_1 \in S_1^T \Rightarrow \) compute \( T_2 = T_1 - t_1 = 57 - 20 = 37 \) and \( E_2 = E_1 = 45 \).

**Assignment decision**
- \( T_2 < E_2 \Rightarrow S_2^E = S_1^E + \{ J_2 \} = \{ J_1, J_2 \} \) and \( S_2^T = S_1^T = \{ J_1 \} \).

**Step 3:** Consider job \( J_3 \) with processing time \( t_3 = 15 \).
- Previous job, \( J_2 \in S_2^E \Rightarrow \) compute \( T_3 = T_2 = 37 \), and \( E_3 = E_2 - t_2 = 45 - 18 = 27 \).

**Assignment decision**
- \( T_3 > E_3 \Rightarrow S_3^E = S_2^E + \{ J_3 \} = \{ J_1, J_2, J_3 \} \) and \( S_3^T = S_2^T = \{ J_2 \} \).

**Step 4:** Consider job \( J_4 \) with processing time \( t_4 = 10 \).
- Previous job, \( J_3 \in S_3^T \Rightarrow \) compute \( T_4 = T_3 - t_3 = 37 - 15 = 22 \) and \( E_4 = E_3 = 27 \).

**Assignment decision**
- \( T_4 < E_4 \Rightarrow S_4^E = S_3^E + \{ J_4 \} = \{ J_1, J_2, J_3, J_4 \} \) and \( S_4^T = S_3^T = \{ J_1, J_3 \} \).

**Step 5:** Consider job \( J_5 \) with processing time \( t_5 = 9 \).
- Previous job, \( J_4 \in S_4^E \Rightarrow \) compute \( T_5 = T_4 = 22 \) and \( E_5 = E_4 - t_4 = 27 - 10 = 17 \).

**Assignment decision**
- \( T_5 > E_5 \Rightarrow S_5^E = S_4^E + \{ J_5 \} = \{ J_1, J_2, J_3, J_4, J_5 \} \) and \( S_5^T = S_4^T = \{ J_2, J_4 \} \).

**Step 6:** Consider job \( J_6 \) with processing time \( t_6 = 8 \).
- Previous job, \( J_5 \in S_5^T \Rightarrow \) compute \( T_6 = T_5 - t_5 = 22 - 9 = 13 \) and \( E_6 = E_5 = 17 \).

**Assignment decision**
- \( T_6 < E_6 \Rightarrow S_6^E = S_5^E + \{ J_6 \} = \{ J_1, J_2, J_3, J_4, J_5, J_6 \} \) and \( S_6^T = S_5^T = \{ J_1, J_3, J_5 \} \).

**Step 7:** Consider job \( J_7 \) with processing time \( t_7 = 7 \).
- Previous job, \( J_6 \in S_6^E \Rightarrow \) compute \( T_7 = T_6 = 13 \) and \( E_7 = E_6 - t_6 = 17 - 8 = 9 \).

**Assignment decision**
- \( T_7 > E_7 \Rightarrow S_7^E = S_6^E + \{ J_7 \} = \{ J_1, J_2, J_3, J_4, J_5, J_6, J_7 \} \) and \( S_7^T = S_6^T = \{ J_2, J_4, J_6 \} \).

**Step 8:** Consider job \( J_8 \) with processing time \( t_8 = 6 \).
- Previous job, \( J_7 \in S_7^T \Rightarrow \) compute \( T_8 = T_7 - t_7 = 13 - 7 = 6 \) and \( E_8 = E_7 = 9 \).
Assignment decision

- $T_8 < E_8 \rightarrow S^E_8 = S^E_7 + \{J_8\} = \{J_2, J_4, J_6, J_8\}$, and
  
  $S^T_8 = S^T_7 = \{J_1, J_3, J_5, J_7\}$. 

Step 9: Consider job $J_9$ with processing time $t_9 = 5$.

- Previous job, $J_9 \in S^E_8 \rightarrow$ compute $T_9 = T_8 = 6$
  
  and $E_9 = E_8 - t_8 = 9 - 6 = 3$.

Assignment decision

- $T_9 > E_9 \rightarrow S^T_9 = S^T_8 + \{J_9\} = \{J_1, J_3, J_5, J_7, J_9\}$, and
  
  $S^E_9 = S^E_8 = \{J_2, J_4, J_6, J_8\}$.

Step 10: Consider job $J_{10}$ with processing time $t_{10} = 4$.

- Previous job, $J_9 \in S^T_9 \rightarrow$ compute $T_{10} = T_9 - t_9 = 6 - 5 = 1$
  
  and $E_{10} = E_9 = 3$.

Assignment decision

- $T_{10} < E_{10} \rightarrow S^T_{10} = S^E_{10} = S^E_9 + \{J_{10}\} = \{J_2, J_4, J_6, J_8, J_{10}\}$, and
  
  $S^T_{10} = S^T_9 = \{J_1, J_3, J_5, J_7, J_9\}$.

End of iteration since all jobs have been assigned to either $S^E_{10}$ or $S^T_{10}$.

Observe that jobs in $S^E_{10} = \{J_2, J_4, J_6, J_8, J_{10}\}$ are in non-increasing order (as required), but jobs in $S^T_{10} = \{J_1, J_3, J_5, J_7, J_9\}$ are not in the required non-decreasing order. Thus, rewrite $S^T_{10} = \{J_9, J_7, J_5, J_3, J_1\}$ which is now in the required non-decreasing order. The optimal schedule is, therefore, given by

$S^* = \{S^E_{10}, S^T_{10}\} = \{J_2, J_4, J_6, J_8, J_{10}, J_9, J_7, J_5, J_3, J_1\}$,

which can be tabulated as

<table>
<thead>
<tr>
<th>$J_i$</th>
<th>$J_2$</th>
<th>$J_4$</th>
<th>$J_6$</th>
<th>$J_8$</th>
<th>$J_{10}$</th>
<th>$J_9$</th>
<th>$J_7$</th>
<th>$J_5$</th>
<th>$J_3$</th>
<th>$J_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>18</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>$c_i$</td>
<td>18</td>
<td>28</td>
<td>36</td>
<td>42</td>
<td>46</td>
<td>51</td>
<td>58</td>
<td>67</td>
<td>82</td>
<td>102</td>
</tr>
<tr>
<td>$</td>
<td>c_i - D</td>
<td>$</td>
<td>27</td>
<td>17</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>13</td>
<td>22</td>
</tr>
</tbody>
</table>

Giving $F^* = \sum_{i=1}^{10} | c_i - D | = 192$ (penalty days).

For comparative purposes, we compute a few selected schedules such as

- $S_1 = \{J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8, J_9, J_{10}\}$,

<table>
<thead>
<tr>
<th>$J_i$</th>
<th>$J_1$</th>
<th>$J_2$</th>
<th>$J_3$</th>
<th>$J_4$</th>
<th>$J_5$</th>
<th>$J_6$</th>
<th>$J_7$</th>
<th>$J_8$</th>
<th>$J_9$</th>
<th>$J_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>20</td>
<td>18</td>
<td>15</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>$c_i$</td>
<td>20</td>
<td>38</td>
<td>53</td>
<td>63</td>
<td>72</td>
<td>80</td>
<td>87</td>
<td>93</td>
<td>98</td>
<td>102</td>
</tr>
<tr>
<td>$</td>
<td>c_i - D</td>
<td>$</td>
<td>25</td>
<td>7</td>
<td>8</td>
<td>18</td>
<td>27</td>
<td>35</td>
<td>42</td>
<td>48</td>
</tr>
</tbody>
</table>

Giving $F_1 = \sum_{i=1}^{10} | c_i - D | = 320$ (penalty days),
\[ S_2 = \{S_{10}^E, S_{10}^E\} = \{J_9, J_7, J_5, J_3, J_1, J_2, J_4, J_6, J_8, J_{10}\}, \]

<table>
<thead>
<tr>
<th>( J_i )</th>
<th>( J_9 )</th>
<th>( J_7 )</th>
<th>( J_5 )</th>
<th>( J_3 )</th>
<th>( J_1 )</th>
<th>( J_2 )</th>
<th>( J_4 )</th>
<th>( J_6 )</th>
<th>( J_8 )</th>
<th>( J_{10} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_i )</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>15</td>
<td>20</td>
<td>18</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>( c_i )</td>
<td>5</td>
<td>12</td>
<td>21</td>
<td>36</td>
<td>56</td>
<td>74</td>
<td>84</td>
<td>92</td>
<td>98</td>
<td>102</td>
</tr>
<tr>
<td>(</td>
<td>c_i - D</td>
<td>)</td>
<td>40</td>
<td>33</td>
<td>24</td>
<td>9</td>
<td>11</td>
<td>29</td>
<td>39</td>
<td>47</td>
</tr>
</tbody>
</table>

\[ F_2 = \sum_{i=1}^{10} |c_i - D| = 342 \text{ (penalty days)}, \]

\[ S_3 = \{J_{10}, J_9, J_8, J_7, J_6, J_5, J_4, J_3, J_2, J_1\}, \]

<table>
<thead>
<tr>
<th>( J_i )</th>
<th>( J_{10} )</th>
<th>( J_9 )</th>
<th>( J_8 )</th>
<th>( J_7 )</th>
<th>( J_6 )</th>
<th>( J_5 )</th>
<th>( J_4 )</th>
<th>( J_3 )</th>
<th>( J_2 )</th>
<th>( J_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_i )</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>15</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>( c_i )</td>
<td>4</td>
<td>9</td>
<td>15</td>
<td>22</td>
<td>30</td>
<td>39</td>
<td>49</td>
<td>64</td>
<td>82</td>
<td>102</td>
</tr>
<tr>
<td>(</td>
<td>c_i - D</td>
<td>)</td>
<td>41</td>
<td>36</td>
<td>30</td>
<td>23</td>
<td>15</td>
<td>6</td>
<td>4</td>
<td>19</td>
</tr>
</tbody>
</table>

\[ F_3 = \sum_{i=1}^{10} |c_i - D| = 268 \text{ (penalty days)}. \]

As can be seen, all the above schedules give the sum of total earliness and tardiness costs more than 192 (penalty days).

**CONCLUSION**

In this paper we have presented a simple heuristic algorithm outlining the procedure for minimizing the sum of total earliness and tardiness costs in an \( n \)-job single machine scheduling problem with common due date. The algorithm involves only \( n \) iterations and is computationally economical for large problems and manually feasible for small problems. We illustrate for a 10-job problem. However, it is assumed that the unit earliness and tardiness costs are constant for all jobs.

What is presented here is just the tip of a large iceberg. Future research may focus on studying similar models in multi-machine environment and try to identify easily solvable special cases. The world of job-machine scheduling is almost endless. For the last fifty years, more than 1200 papers on various aspects of the problem have been published in the operational research and management science literatures. The advent of just-in-time system and development in supply chain management, internet and e-commerce has created new and complex scheduling problems to the existing problems that we have just begun to understand.

**REFERENCES**


