Approximation Algorithms to Solve Simultaneous Multicriteria Scheduling Problems

Ali Musaddak Delphi

Department of Mathematics, College of Basic Education, University of Misan, Iraq

ABSTRACT

In this paper, we discuss the multicriteria scheduling problem on single machine to minimize total completion time, total earliness, and maximum tardiness or earliness. We proposed approximation algorithm to solve the first problem \(1//F(\sum C_i,T_{max},\sum E_i)\), and approximation algorithm to solve the second problem \(1//F(\sum C_i,E_{max},\sum E_i)\) to find the set of efficient solutions. Also for the problem \(1//F(\sum C_i,T_{max},E_{max},\sum E_i)\), we proposed constructive approximation algorithm based on the previous two approximation algorithms to find some of the efficient solutions for the four criteria optimality.

Keywords: Scheduling problems, Single machine, Multicriteria scheduling, Efficient algorithms.

1. Introduction

We address the following single machine multicriteria scheduling problem. A set of \(n\) independent jobs has to be scheduled on single machine, which can handle only one job at a time. The machine is assumed to be continuously available from time zero and awards. Each job \(i\) (\(i=1,...,n\)) requires \(p_i\) time units to be processed on the machine and due date \(d_i\) for ideally completing job \(i\). All jobs are available for processing at time zero. A schedule \(\pi=(\pi(1),...\pi(n))\) specifies for each job when it is executed while observing the machine availability constraints and the schedule produce a completion time \(C_{\pi(i)}=\sum_{j<i}p_{\pi(j)}\) for each job \(i\). Some of the performance measures often used in scheduling are, sum of completion time \(\sum C_i\), total earliness \(\sum E_i\), total tardiness \(\sum T_i\), maximum lateness \(L_{max} = \text{max}\{C_i - d_i,0\}\), maximum earliness \(E_{max} = \text{Max}\{E_i\}\), earliness \(E_i = \text{Max}\{d_i - C_i,0\}\), and maximum tardiness \(T_{max} = \text{Max}\{T_i\}, T_i = \text{Max}\{C_i - d_i,0\}\). Throughout this paper, we use the three field notation scheme \(\alpha/\beta/\gamma\) introduced by Graham et.al., [1] to denote the scheduling problem under consideration. Most of the work done in bicriteria problems has been on the single machine bicriteria problems. Many researchers used branch and bound (BAB) type of algorithms to solve the problem while some approaches have utilized dynamic programming (DP) algorithms. In this paper, we will use the following sequencing rules:

1. SPT: Jobs are sequenced in non-decreasing order of processing times, (this rule is well known to minimize \(\sum C_i\)) for \(1//\sum C_i\) problem. [1]

2. EDD: Jobs are sequenced in non-decreasing order of due dates, (this rule is well known to minimize \(T_{max}\)) for \(1//T_{max}\) problem. [2]

3. MST: Jobs are sequenced in non-decreasing order of slack time, (non-decreasing order of \(S_j = d_j - p_j\)) , (this rule is well known to minimize \((1//E_{max})\) problem subject to no machine idle time). [2]

2. Basic concept of multicriteria scheduling

As to bicriteria scheduling problems, two different criteria are considered together this can be accomplished in a number of ways. One approach is to minimize the less important criterion, subject to the restriction that the most important criterion optimized. The two criteria are assumed to be prioritized as primary and secondary with the objective of finding the best schedule for the secondary criterion among all alternative optimal schedules for the primary criterion. The optimal solution obtained from this approach is called a hierarchical (lexicographical) schedule, and the problem is denoted by \(1//\text{Lex}(\gamma^1,\gamma^2)\) where \(\gamma^1\) is the primary criterion and \(\gamma^2\) is the secondary criterion. The second approach is called simultaneous optimization this approach unlike hierarchical
minimization approach, where the performance criteria \( \gamma^1, \gamma^2 \) are indexing in the same order of important, and we have two type of problems the first one is \( 1/1(F(\gamma^1), \gamma^2) \) and the second one \( 1/1(\gamma^1 + \gamma^2) \).

**Definition (1)**[3]: A measure of performance is said to be regular if it is a non-decreasing function of job completion times. Examples of regular measures are job flow time \( \sum C_i \), schedule makespan \( C_{\text{max}} \) and tardiness based performance measures.

**Definition (2)**[3]: A non-regular performance measure is usually not a monotone function of the job completion times. An example of such a measure is the job earliness.

**Definition (3)**[4]: A feasible solution \( \sigma \in \Pi \) is called efficient if there is no other solution \( \pi \in \Pi \) such that \( \gamma^1(\pi) \leq \gamma^1(\sigma) \) for every \( i = 1, \ldots, k \) and \( \gamma^1(\pi) < \gamma^1(\sigma) \) for some \( j \). If \( \sigma \) is efficient solution, then \( \gamma(\sigma) \) is called a non-dominated point. The set of all efficient solutions \( \sigma \in \Pi \) is called efficient set and the set of all non-dominated points \( \gamma(\sigma) \) is called non-dominated set or Pareto front.

3. The approximation algorithms and simultaneous problems

There are many algorithms that can be used for solving multicriteria scheduling problems, which is to find the efficient solutions or at least approximation to it. The running time for the algorithm often increases with the increase of the instance size. The purpose of any algorithm process is to find, for each instance a feasible solution called optimal, that minimize the objective function. This usual meaning of the optimum makes no sense in the multicriteria case because it doesn’t exist, in most of the cases, a solution optimizing all objectives simultaneously. Hence we search for feasible solutions yielding the best compromise among objectives that constitutes a so called efficient solution set. These efficient solutions that cannot be improved in one objective without decreasing their performance in at least one of the others. It is clear that this efficient solutions set is difficult to find. Therefore, it could be preferable to have an approximation to that set in a reasonable amount of time. In this section, we shall try to find efficient (Pareto optimal) solutions for the \( 1/1(F(\sum C_i, T_{\text{max}}, \sum E_i)) \), \( 1/1(F(\sum C_i, C_{\text{max}}, \sum E_i)) \) and \( 1/1(F(\sum C_i, T_{\text{max}}, E_{\text{max}}, \sum E_i), \) which are NP-hard multicriteria scheduling problems.

Let the first problem \( 1/1(F(\sum C_i, T_{\text{max}}, \sum E_i)) \) denoted by \( (P_1) \). This multicriteria scheduling problem \( (P_1) \) has the mathematical form:

\[
\text{Min} (\sum C_i, T_{\text{max}}, \sum E_i) \text{ s.t. } \begin{align*}
C_i &\geq P_i \\
C_i &\geq C_{i-1} + p_i \\
T_i &\geq C_i - d_i, E_i \geq d_i - C_i \\
T_i &\geq 0, E_i \geq 0
\end{align*} \quad (P_1)
\]

For this problem \( (P_1) \), \( (\sum C_i, T_{\text{max}}) \) and \( (\sum E_i) \) has the same important objective function and should be efficient for any feasible schedules. The following approximation algorithm (SCTSE) gives the efficient solutions for the problem \( (P_1) \).

**Algorithm (1): approximation algorithm (SCTSE)**

*Step (0):* put \( \Delta = \sum P_i, \mathbb{N} = \{1, \ldots, n\}, k = n, 0 = C_{\text{max}} = \sum P_i \) and \( \sigma = (\phi) \).

*Step (1):* calculate \( T_i', B_i' \) for each \( i \in \mathbb{N} \) as follows:

\[
T_i' = \max(C_i - d_i), B_i' = \max(d_i - C_i, 0).
\]

*Step (2):* find job \( J^* \in \mathbb{N}, \) such that \( T_i' \leq \Delta, p_{J^*} > p_i \) for each \( J^*, i \in \mathbb{N} \) and \( T_i' \leq \Delta \), if \( p_{J^*} = p_i \) choose the job with smallest \( T_i' \) and if \( T_{J^*} = T_i \) choose the job with smallest \( B_i \), then assign job \( J^* \) in position \( k \) of \( \sigma = (\sigma(k)), \sigma(k) \), if no job with \( T_{J^*} \leq \Delta \) go to step(3).

*Step (3):* set \( t = t - p_i, \mathbb{N} = \mathbb{N} - (J^*), K = K - 1, \) if \( K > 1 \) go to step(1), otherwise go to step(4).

*Step (4):* compute \( \sum C_{\sigma(1)}, T_{\text{max}}(\sigma) \) and \( \sum E_{\sigma(1)} \) for sequence jobs \( \sigma = (\sigma(1), \ldots, \sigma(\ell)) \), and set \( \Delta = T_{\text{max}} - 1, \mathbb{N} = \{1, \ldots, n\}, k = n, t = C_{\text{max}} = \sum P_i, \sigma = (\phi) \), go to step(1).

*Step (5):* stop.
Example (1): Consider the problem \((P_1)\) with the following data:

<table>
<thead>
<tr>
<th>Pi</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Di</td>
<td>10</td>
<td>15</td>
<td>13</td>
<td>11</td>
</tr>
</tbody>
</table>

Hence, efficient schedules for the three criteria NP-hard problem \((P_1)\) of minimizing \(\sum C_i, T_{\text{max}}\) and \(\sum E_i\) are given by the approximation algorithm (SCTSE)

<table>
<thead>
<tr>
<th>Schedules</th>
<th>(\sum C_i)</th>
<th>(T_{\text{max}})</th>
<th>(\sum E_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPT-(1,2,3,4)</td>
<td>43</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>(1,2,4,3)</td>
<td>44</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>(1,3,4,2)</td>
<td>51</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>(1,4,3,2)</td>
<td>52</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Proposition (1): Algorithm (SCTSE) determines almost efficient schedules for \(1//F(\sum C_i, T_{\text{max}}, \sum E_i)\).

Proof: it suffices to show that the algorithm generates a schedule \(\sigma\) that solves the problem \(1/T_{\text{max}} \leq \Delta / F(\sum C_i, \sum E_i)\) and \(1/\sum C_i \leq \sum C_i(\sigma) / F(T_{\text{max}}, \sum E_i)\) simultaneously. Suppose that \(\sigma\) solves \(1/T_{\text{max}} \leq \Delta / F(\sum C_i, \sum E_i)\) and \(\pi\) optimal for \(1/\sum C_i \leq \sum C_i(\sigma) / F(T_{\text{max}}, \sum E_i)\), not \(\sigma\). This leads to that \(T_{\text{max}}(\pi) < T_{\text{max}}(\sigma) \leq \Delta\); hence, \(\pi\) is also feasible for \(1/T_{\text{max}} \leq \Delta / F(\sum C_i, \sum E_i)\), and so, we have \(\sum C_i(\pi) = \sum C_i(\sigma)\). Compare the two schedules starting at the end, let that the first difference occurs at the \(k-th\) position, which is occupied by job \(i\) and \(j\) in \(\sigma\) and \(\pi\) respectively. Since \(T_{\text{max}}(\pi) < \Delta\) and because of the choice of job \(i\) in the algorithm, we have \(p_i \geq p_j\). If \(p_i > p_j\), then \(\pi\) cannot be optimal, as schedule that is obtained by interchanging job \(i\) and \(j\) in \(\pi\) is feasible with respect to the constraint \(T_{\text{max}} \leq \Delta\) and has smaller completion time. Hence, it must be that \(p_i = p_j\). This implies, however that the job \(i\) and \(j\) can be interchanged in \(\pi\) without effecting the cost of the schedule. Repetition of this argument shows that \(\pi\) can be transformed into \(\sigma\) without affecting the cost, and so this a contradiction with the assumption that \(T_{\text{max}}(\pi) < T_{\text{max}}(\sigma)\). Therefore, \(\sigma\) also solves \(1/\sum C_i \leq \sum C_i(\sigma) / F(T_{\text{max}}, \sum E_i)\); hence, \(\sigma\) is efficient schedules for \(1//F(\sum C_i, T_{\text{max}}, \sum E_i)\).

Let the second problem \(1//F(\sum C_i, E_{\text{max}}, \sum E_i)\) denoted by \((P_2)\). This multicriteria scheduling problem \((P_2)\) has the mathematical form

\[
\begin{align*}
\text{Min} & (\sum C_i, E_{\text{max}}, \sum E_i) \\
\text{s.t.} & \ C_i \geq E_i \\
& \ C_i = C_{i-1} + p_i \quad i = 2, \ldots, n \\
& \ E_i \geq d_i - C_i \quad i = 1, \ldots, n \\
& \ E_i \geq 0 \quad i = 1, \ldots, n 
\end{align*}
\]

\((P_2)\)
For this problem \((P_2)\), \((\sum C_i, E_{\text{max}})\) and \((\sum E_i)\) has the same important objective function and should be efficient for any feasible schedules. The following approximation algorithm (SCESE) gives the efficient solutions for the problem \((P_2)\).

### Algorithm (2): approximation algorithm (SCESE)

- **Step (0):** Input \(C_{\text{max}}\), and set 
  \[ \sigma_n = C_{\text{max}}, h = 2, N = \{1, \ldots, n\}, k = 1, t = C_{\text{max}} = \sum p_i, \text{ put } \Delta = E_{\text{max}} < \text{SPT} - 1, \text{ and compute } E_{\text{max}} < \text{MST}. \]
- **Step (1):** Calculate \(E_i, S_i\) for each \(i \in N\) as follows: 
  \[ E_i = \max(d_i - C_i, 0), S_i = d_i - p_i \]
- **Step (2):** Find job \(J^* \in N\), such that \(E_{J^*} \leq b, p_{J^*} < p_i\) for each \(J^* \in N\) and \(E_i \leq \Delta\), if \(p_{J^*} = p_i\) choose the job with smallest \(E_{J^*}\), if \(E_{J^*} = E_i\) choose the job with smallest slack time \(S_{J^*}\) and if \(h = n\) choose the job with smallest \(E_{J^*}\), such that \(E_{J^*} \leq b, p_{J^*} < p_i\) for each \(J^* \in N\) and \(E_i \leq \Delta\), then assign job \(J^*\) in position \(k\) of 
  \[ \sigma_n = (\sigma, \sigma[k]), \]
- **Step (3):** Set \(N = N - (J^*), K = K + 1\), if \(K < n\) go to step (1), otherwise go to step (2).
- **Step (4):** Compute \(\sum C_{\text{seq}}(\sigma)\) and \(\sum B_{\text{seq}}(\sigma)\) for sequence jobs \(\sigma_n = (\sigma(1), \ldots, \sigma(n))\), and set 
  \[ \Delta = E_{\text{max}} - 1, N = \{1, \ldots, n\}, k = 1, t = C_{\text{max}} = \sum p_i, h = k + 1 \text{ and } \sigma_n = \sigma_{n, 2} \text{ if } \Delta < E_{\text{max}} < \text{MST} \text{ go to step (5)}, \text{ otherwise go to step (1).} \]
- **Step (5):** Stop.

Modify this procedure as follows: break tie in step (2) by selecting the job with smallest slack time \(S_{J^*}\) for the last sequence such that \(E_{J^*} \leq \Delta, p_{J^*} < p_i\) for each \(J^* \in N\) and \(E_i \leq \Delta\), where by our computations, we found the number of efficient solutions which can be found by algorithm (2) and modified procedure are determined at least by \((n + 1)\) efficient solutions for each number of jobs \((n)\). As shown in the following example:

**Example (2):** Consider the problem \((P_2)\) as in the previous example data:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_i)</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>(D_i)</td>
<td>10</td>
<td>15</td>
<td>13</td>
<td>11</td>
</tr>
</tbody>
</table>

Hence, efficient schedules for the three criteria NP-hard problem \((P_2)\) of minimizing \(\sum C_i, E_{\text{max}}\) and \(\sum E_i\) are given by the approximation algorithm (SCESE)

<table>
<thead>
<tr>
<th>Schedules</th>
<th>(\sum C_i)</th>
<th>(E_{\text{max}})</th>
<th>(\sum E_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPT-(2,1,3,4)</td>
<td>43</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>(1,2,3,4)</td>
<td>44</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>(1,3,2,4)</td>
<td>47</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>(4,2,1,3)</td>
<td>51</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>(4,1,2,3)</td>
<td>52</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

**Proposition (2):** Algorithm (SCESE) determines almost efficient solutions for \(1 / F(\sum C_i, E_{\text{max}}, \sum E_i)\).

**Proof:** It suffices to show that the algorithm generates a schedule \(\sigma\) that solves the problem \(1 / E_{\text{max}} \leq \Delta / F(\sum C_i, \sum E_i)\) and \(1 / \sum C_i \leq \sum C_i(\sigma) / F(E_{\text{max}}, \sum E_i)\) simultaneously. Let \(\sigma\) solves \(1 / E_{\text{max}} \leq \Delta / F(\sum C_i, \sum E_i)\) and \(\pi\) optimal for \(1 / \sum C_i \leq \sum C_i(\sigma) / F(E_{\text{max}}, \sum E_i)\).
not $\sigma$. This leads to that $E_{\max}(\pi) < E_{\max}(\sigma) \leq \Delta$; hence, $\pi$ is also feasible for $1/E_{\max} \leq \Delta / F(\sum C_i, \sum E_i)$, and so we have $\sum C_i(\pi) = \sum C_i(\sigma)$. Compare the two schedules starting at the end, let that the first difference occurs at the $k$-th position, which is occupied by job $i$ and $j$ in $\sigma$ and $\pi$ respectively. Since $E_{\max}(\pi) < \Delta$ and because of the choice of job $i$ in the algorithm, we have $p_i \geq p_j$. If $p_i > p_j$, then $\pi$ cannot be optimal, as schedule that is obtained by interchanging job $i$ and $j$ in $\pi$ is feasible with respect to the constraint $E_{\max} \leq \Delta$ and has smaller completion time. Hence, it must be that $p_i = p_j$. This implies, however that the job $i$ and $j$ can be interchanged in $\pi$ without effecting the cost of the schedule. Repetition of this argument shows that $\pi$ can be transformed into $\sigma$ without affecting the cost, and so this a contradiction with the assumption that $E_{\max}(\pi) < E_{\max}(\sigma)$. Therefore, $\sigma$ also solves $1/F(\sum C_i, E_{\max}, \sum E_i)$. Hence, $\sigma$ is efficient schedules for $1/F(\sum C_i, E_{\max}, \sum E_i)$.

Now, we present a approximation algorithm to find some of the efficient solutions for NP-hard problem $(1/F(\sum C_i, \max E, \sum E_i))$, when all the criteria $\sum C_i, \max E, \sum E_i$ are of simultaneous interest in problem $(P)$. i.e., some of the efficient combinations of $(\sum C_i, \max E, \sum E_i)$. This multicriteria scheduling problem $(P)$ has the mathematical form:

$$\text{Min}(\sum C_i, \max T, \max E, \sum E_i)$$

s.t.

$$C_i \geq p_i, \quad i = 1, \ldots, n$$
$$C_i = C_{i-1} + p_i, \quad i = 2, \ldots, n$$
$$T_i \geq C_i - d_i, \quad E_i \geq c_i - C_i, \quad i = 1, \ldots, n$$
$$T_i \geq 0, \quad E_i \geq 0, \quad i = 1, \ldots, n$$

$(P)$

The main idea of the following approximation algorithm is depending on the previous approximation algorithms (SCTSE and SCESE) and the definition of efficient solutions.

**Algorithm (3): approximation algorithm (SCTSE)**

*Step (0):* put $\Delta = \sum p_i, N = \{1, \ldots, n\}, k = n, t = C_{\max} = \sum p_i$ and $\sigma = (\phi)$.

*Step (1):* calculate $T_i, E_i, S_i$ for each $i \in N$ as follows:

$$T_i = \max(C_i - d_i, 0), E_i = \max(d_i - C_i, 0), S_i = d_i - p_i.$$  

*Step (2):* find job $j^* \in N$, such that $T_{j^*} \leq \Delta, p_{j^*} > p_i$ for each $j^*, i \in N$ and $T_i \leq \Delta$, if $p_{j^*} = p_i$, choose the job with smallest $T_{j^*}$ and if $T_{j^*} < T_i$, choose the job with smallest $E_{j^*}$, then assign job $j^*$ in position $k$ of $\sigma = (\sigma(\ell), \sigma(\ell))$, if no job with $T_{j^*} \leq \Delta$ go to step(5).

*Step (3):* set $t = t - p_{j^*}, N = N - \{j^*\}, k = k - 1$, if $k > 1$ go to step(1), otherwise go to step(4).

*Step (4):* compute $\sum C_{e_{(i)}} T_{\max}(\sigma), E_{\max}(\sigma)$ and $\sum E_{e_{(i)}}$ for sequence jobs $\sigma = (\sigma(1), \ldots, \sigma(n))$, and set $\Delta = T_{\max} - 1, N = \{1, \ldots, n\}, k = n, t = C_{\max} = \sum p_i, \sigma = (\phi)$, go to step(1).

*Step (5):* put $\sigma_h = \sigma_{SP}, h = 1, N = \{1, \ldots, n\}, k = 1, t = C_{\max} = \sum p_i, \Delta = E_{\max}(\text{SPT}) - 1$, and compute $E_{\max}(\text{MST}).$

*Step (6):* find job $j^* \in N$, such that $E_{j^*} \leq \Delta, p_{j^*} < p_i$ for each $j^*, i \in N$ and $E_i \leq \Delta$, if $p_{j^*} = p_i$, choose the job with smallest $E_{j^*}$, if $E_{j^*} = E_{j^*}$, choose the job with smallest slack time $S_{j^*}$ and if $h = n$ choose the job with smallest $E_{j^*}$, such that $E_{j^*} \leq \Delta, p_{j^*} < p_i$ for each $j^*, i \in N$ and $E_i \leq \Delta$, then assign job $j^*$ in position $k$ of $\sigma = (\sigma(\ell), \sigma(\ell)).$

*Step (7):* set $N = N - \{j^*\}, K = K + 1$, if $K < n$ go to step(1), otherwise go to step(5).
Example (3): Consider the problem \((P)\) with the following data:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p_i)</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>(d_i)</td>
<td>15</td>
<td>10</td>
<td>8</td>
<td>11</td>
</tr>
</tbody>
</table>

Hence, efficient schedules for the four criteria NP-hard problem \((P)\) of minimizing \(\sum C_i, T_{\text{max}}, E_{\text{max}}, \sum E_i\) are given by the approximation algorithm (SCTESE).

<table>
<thead>
<tr>
<th>Schedules</th>
<th>(\sum C_i)</th>
<th>(T_{\text{max}})</th>
<th>(E_{\text{max}})</th>
<th>(\sum E_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPT=(3,1,2,4)</td>
<td>31</td>
<td>7</td>
<td>12</td>
<td>20</td>
</tr>
<tr>
<td>(3,1,4,2)</td>
<td>34</td>
<td>8</td>
<td>12</td>
<td>19</td>
</tr>
<tr>
<td>(3,2,1,4)</td>
<td>35</td>
<td>7</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>(2,3,1,4)</td>
<td>40</td>
<td>7</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>(3,2,4,1)</td>
<td>42</td>
<td>5</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>(2,3,4,1)</td>
<td>47</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>(4,3,1,2)</td>
<td>49</td>
<td>8</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>(4,3,2,1)</td>
<td>53</td>
<td>6</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**Proposition (3):** Algorithm (SCTESE) determines almost efficient schedules for \(\text{SCTESE} \leq F(\sum C_i, T_{\text{max}}, E_{\text{max}}, \sum E_i)\).

**Proof:** The prove follows immediately by using proposition(1) and proposition(2).

4. Conclusion

In this paper, we conclude that the three approximation algorithms found almost efficient schedules for the three problems optimality compared with complete enumeration method.

References

AUTHOR
Ali Musaddak Delphi received the B.Sc. in Mathematics from Mathematics Department, College of Science, Al-Mustansiriyah University, Iraq, from the period of 2004-2007. M.Sc. in Mathematics from Mathematics Department, College of Science, Al-Mustansiriyah University, Iraq, from 2009-2011. He is now assistant lecturer in Mathematics Department, College of basic education, Misan University, Iraq. He is interesting in scheduling theory, approximation theory and functional analysis.