Dual Objective: Cmax and $\sum C_j$

Minimizing Cmax subject to minimum $\sum C_j$

**Thm:** $P_2 \parallel C_{\text{max}}$ ($\sum C_j$ is minimum) is NP-hard

**Proof:**
(1) in NP (Obvious)

(2) a. Reduction

Reduction from even-odd partition

Even-Odd Partition Problem:
Given a list $A = (a_1, a_2, \ldots, a_{2n})$ of $2n$ integers, where $a_1 < a_2 < \ldots < a_{2n}$. Can $A$ be partition into $A_1$ and $A_2$, with both $A_1$ and $A_2$ have exactly one element of each pair of $a_{2i-1}$ and $a_{2i}$ for all $1 \leq i \leq n$, such that $\sum_{(aj \in A_1)} aj = \sum_{(aj \in A_2)} aj$?

Let $a_1 < a_2 < \ldots < a_{2n}$ be an instance of even-odd partition. Construct $2n$ job, with $P_j$ of job $j$ equal to $aj$.

Threshold $\beta = \frac{1}{2} \sum_{(j=1 \text{ to } 2n)} aj$

(2) b. if and only if
There is a schedule with $C_{\text{max}} \leq \beta$ iff there is a solution for even-odd partition.
P | prmp | Cmax (\(\sum C_j\) is minimum) can be solved in polynomial time.

O(n log n) time


1. Assume n is a multiple of m. Otherwise, introduce dummy jobs with zero processing times.
2. Sort the jobs in ascending order of processing times.
3. Schedule the first n-m jobs by strict SPT mode.
4. Compute OFT as follows.

Assume \(f_1 \leq f_2 \leq \ldots \leq f_m\), where \(f_i\) is the finishing time of machine \(i\)
   
   \begin{align*}
   (i) & \quad f_1 + P_n \\
   (ii) & \quad \frac{1}{2} (f_1 + f_2 + P_n + P_{n-1}) \\
   (iii) & \quad \frac{1}{3} (f_1 + f_2 + f_3 + P_n + P_{n-1} + P_{n-2}) \\
   \quad & \quad \ldots \\
   (m) & \quad \frac{1}{m} (f_1 + f_2 + \ldots + f_m + P_n + P_{n-1} + \ldots + P_{n-m+1})
   \end{align*}

OFT is the maximum of the above \(m\) quantities.

<table>
<thead>
<tr>
<th>f₁</th>
<th>f₂</th>
<th>f₃</th>
<th>OFT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f_{m-1}</td>
<td>f_m</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5. Schedule the jobs \( n, n-1, n-2, \ldots, n-m+1 \), in that order as follows.

**Rule 1:** If the job can fit completely in the machine with least capacity, then schedule the job into the machine and delete the machine from consideration.

**Rule 2:** If the job can fit perfectly into one machine without any idle time, then schedule the job on that machine and delete the machine from consideration.

else **Rule 3:** There must be an index \( i \) such that \((OFT - f_i) > P_j\) and \((OFT - f_{i+1}) < p_j\).

Schedule \((OFT - f_{i+1})\) amount of job \( j \) on machine \( i+1 \) and the remainder on machine \( i \). Delete machine \( i+1 \) from consideration.

**Illustrate with an example**