Chapter 8

Operations Scheduling
Operations Scheduling
Scheduling is
- the process of organizing, choosing and timing resource usage to carry out all the activities necessary to produce the desired outputs at the desired times, while satisfying a large number of time and relationship constraints among the activities and the resources (Morton and Pentico, 1993).

Schedule specifies
- the time each job starts and completes on each machine, as well as any additional resources needed.

A Sequence is
- a simple ordering of the jobs.
Determining a best sequence
32 jobs on a single machine
32! Possible sequences approx. 2.6x10^{35}
  - suppose a computer could examine one billion sequences per second
  - it would take 8.4x10^{15} centuries
real life problems are much more complicated
Scheduling theory helps to
  - classify the problems
  - identify appropriate measures
  - develop solution procedures
Algorithmic complexity

- An efficient algorithm is one whose effort of any problem instance is bounded by a polynomial in the problem size, e.g. # of jobs.
- Minimal spanning tree can be solved in at most $n^2$ iterations.
- $n$: number of edges.
- $O(n^2)$.

- If effort is exponential $O(2^n)$ the algorithm is not efficient.
- Branch and bound algorithm for 0/1 variables.

- NP-hard problems: no exact algorithm in polynomial time is known. e.g. Traveling salesman problem.
- Heuristics are usually polynomial algorithms tailored to the specific problem structure.
Operations Scheduling

Graph showing the comparison of $n^2$ and $2^n$ functions.
Operations Scheduling

- **Scheduling Theory (Background)**
- **Jobs are**
  - activities to be done
  - processing time known
  - in general continuously processed until finished (preemption not allowed)
  - due date
  - release date
  - precedence constraints
  - sequence dependent setup time
  - processed by at most one-machine at the same time
Operations Scheduling

- **Machines (resources)**
  - single machine, parallel machines
  - flow shop:
    - each job must be processed by each machine exactly once
    - all jobs have the same routing
    - a job cannot begin processing on the second machine until it has completed processing on the first
  - assembly line
  - job shop:
    - each job may have a unique routing
  - open shops:
    - job shops in which jobs have no specific routing
    - re-manufacturing and repair
Operations Scheduling

**Measures**
- profit, costs
- it is difficult to relate a schedule to profit and cost
- **regular measure** is a function of completion time
  - function only increases if at least one completion time in schedule increases

- $n =$ number of jobs to be processed
- $m =$ number of machines
- $p_{ik} =$ time to process job $i$ on machine $k$
- $r_i =$ release date of job $i$
- $d_i =$ due date of job $i$
- $w_i =$ weight of job $i$ relative to the other jobs
Operations Scheduling

- $C_i$ = the completion time
- $F_i = C_i - r_i$, the flowtime
- $L_i = C_i - d_i$, lateness of job $i$
- $T_i = \max\{0, L_i\}$, tardiness of job $i$
- $E_i = \max\{0, -L_i\}$, earliness of job $i$

- $\delta_i = 1$, if job $i$ is tardy ($T_i > 0$)
- $\delta_i = 0$, if job $i$ is on time ($T_i = 0$)

$C_{\text{max}} = \max_{i=1,n} \{C_i\}$, makespan

$L_{\text{max}} = \max_{i=1,n} \{L_i\}$, maximum lateness

$T_{\text{max}} = \max_{i=1,n} \{T_i\}$, maximum tardiness
Operations Scheduling

- **Common proxy objectives**
  - total flowtime
  - total tardiness
  - makespan
  - maximum tardiness
  - number of tardy jobs
  - if not all jobs are equally important weights should be introduced

- minimizing total completion time is equivalent to minimizing total flowtime or minimizing total tardiness
Operations Scheduling

- **Algorithms:**
  - exact algorithms often based on (worst case scenario) enumeration (e.g. Branch and Bound, Dynamic Programming)
  - heuristic algorithm judged by quality (difference to the optimal solution) and efficacy (computational effort)
  - worst-case bounds are desirable to motivate use of a certain heuristic
Operations Scheduling

Consider the following four-job, three-machine job-shop scheduling problem:

<table>
<thead>
<tr>
<th>Job</th>
<th>Op.1</th>
<th>Op.2</th>
<th>Op.3</th>
<th>Release Date</th>
<th>Due date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4/1</td>
<td>3/2</td>
<td>2/3</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>4/1</td>
<td>4/3</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>3/3</td>
<td>2/2</td>
<td>3/1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>3/2</td>
<td>3/3</td>
<td>1/1</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

Assume the following sequences:
- 2-1-4-3 on M1
- 2-4-3-1 on M2
- 3-4-2-1 on M3
Operations Scheduling

Gantt Chart (machine oriented)
**Operations Scheduling**

\[ C_1 = 14, C_2 = 11, C_3 = 13, C_4 = 10 \]

The makespan is

\[ C_{\text{max}} = \max \{ C_1, C_2, C_3, C_4 \} = \max \{ 14, 11, 13, 10 \} = 14 \]

The total flowtime is

\[ \sum_i F_i = 14 + 11 + 13 + 10 = 48 \]
Operations Scheduling

The lateness and the tardiness of a job:

\[ L_1 = 14 - 16 = -2 \quad T_1 = \max \{0, -2\} = 0 \]
\[ L_2 = 11 - 14 = -3 \quad T_2 = \max \{0, -3\} = 0 \]
\[ L_3 = 13 - 10 = 3 \quad T_3 = \max \{0, 3\} = 3 \]
\[ L_4 = 10 - 8 = 2 \quad T_4 = \max \{0, 2\} = 2 \]

The total lateness is

\[ \sum_i L_i = (-2) + (-3) + 3 + 2 = 0 \]

The total tardiness is

\[ \sum_i T_i = 0 + 0 + 3 + 2 = 5 \]

The maximum tardiness is

\[ T_{\text{max}} = \max \{0, 0, 3, 2\} = 3 \]

Tardy jobs have \( \delta_i = 1 \), so

The number of tardy jobs is

\[ T_1 = 0 \Rightarrow \delta_1 = 0 \]
\[ T_2 = 0 \Rightarrow \delta_2 = 0 \]
\[ T_3 > 0 \Rightarrow \delta_3 = 1 \]
\[ T_4 > 0 \Rightarrow \delta_4 = 1 \]

\[ N_T = 2 \]
Operations Scheduling

- Single Machine Scheduling
- Minimizing Flowtime

- Problem data
  - Job: 1 2 3 4 5
  - p: 4 2 3 2 4

- Sequence: 1-2-3-4-5
- Total Flowtime = ?
- Total Flowtime = np + (n-1)p + ... + p
- F = np + (n-1)p + ... + p

Theorem. SPT sequencing minimizes total flowtime on a single machine with zero release times.

Proof. We assume an optimal schedule is not an SPT sequence.

- $p_i > p_j$
- $TF(S) = TF(B) + (t+p_i) + (t+p_i+p_j) + TF(A)$
- $TF(S') = TF(B) + (t+ p_j) + (t+p_j+p_i) + TF(A)$
- $TF(S) - TF(S') = p_i - p_j > 0$
Operations Scheduling

SPT-rule ⇒ sequence: 2-4-3-1-5

\[ C_1 = 11 \]
\[ C_2 = 2 \]
\[ C_3 = 7 \]
\[ C_4 = 4 \]
\[ C_5 = 15 \]

Total flowtime = total completion time = 39

* SPT rule also minimizes
  - total waiting time
  - mean # of jobs waiting (mean work in progress)
  - total lateness

* Why?
Operations Scheduling

- Minimize weighted Flow-time: \[ \sum_{i=1}^{n} w_i F_i \]

- weighted SPT (WSPT): order ratios \[ \frac{p_i}{w_i} \] (nondecreasing)

- exact algorithm for weighted flow-time with zero release time (completion time)
Operations Scheduling

Weighted Flowtime

WSPT scheduling

\[ w_1 = 1, \ w_2 = 4, \ w_3 = 3, \ w_4 = 1, \ w_5 = 3 \]

the processing-time-to-weight ratio gives: 4; 0,5; 1; 2; 1,33

the WSPT sequence is the following: 2-3-5-4-1

\[ C_1 = 15 \]
\[ C_2 = 2 \]
\[ C_3 = 5 \]
\[ C_4 = 11 \]
\[ C_5 = 9 \]

the value of weighted flowtime is

\[ \sum_{i=1}^{5} w_i F_i = 76 \]
**Operations Scheduling**

- **Maximal Tardiness and Maximal Lateness**
  - due date oriented measure
  - earliest due date sequence (EDD)
  - EDD minimizes
    - Maximal Tardiness and
    - Maximal Lateness

<table>
<thead>
<tr>
<th>Job i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due date</td>
<td>16</td>
<td>10</td>
<td>7</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Proc. Time</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

- EDD-sequence: 5-3-4-2-1
- Tardiness of the jobs is (0, 0, 2, 1, 0)
Operations Scheduling

▲ Number of Tardy Jobs

● Hodgson’s algorithm

● Step 1. Compute the tardiness for each job in the EDD sequence. Set \( N_T = 0 \), and let \( k \) be the first position containing a tardy job. If no job is tardy go to step 4.

● Step 2. Find the job with the largest processing time in positions 1 to \( k \).

\[
\text{Let } p_{[j]} = \max_{i=1,k} p_{[i]} \text{ then } j^* = \lceil j \rceil
\]

● Step 3. Remove job \( j^* \) from the sequence, set \( N_T = N_T + 1 \), and repeat Step 1.

● Step 4. Place the removed \( N_T \) jobs in any order at the end of the sequence.

● This sequence minimizes the number of tardy jobs
Consider the previous example:
EDD-sequence: 5-3-4-2-1

Step 1: The tardiness is (0, 0, 2, 1, 0) ⇒ Job 4 in the third position is the first tardy job;

Step 2: The processing times for jobs 5, 3 and 4 are 4, 3, 2, respectively;
⇒ largest processing time for job 5

Step 3: Remove job 5, goto step 1

Step 1: EDD-sequence is 3-4-2-1; completion times (3, 5, 7, 11) and tardiness (0, 0, 0, 0) ⇒ Go to step 4

Step 4: schedule that minimizes the number of tardy jobs is 3-4-2-1-5 and has only one tardy job: Job 5
Operations Scheduling

- Minimize the weighted number of tardy jobs!
- NP-hard Problem
- Heuristic approach: processing-time-to-weight ratio (not exact!)

Consider the previous example with the following weights:

\[ w_1 = 1, \ w_2 = 4, \ w_3 = 3, \ w_4 = 1, \ w_5 = 3 \]

- EDD-sequence was 5-3-4-2-1
- **Step 1** first tardy job is job 4
- **Step 2** the processing-time-weight-ratio for jobs 5, 3 and 4 are 4/3, 3/3 and 2/1
- **Step 3** Remove job 4
- **Step 1** EDD-sequence is 5-3-2-1 with no tardiness
- **Step 4** new schedule 5-3-2-1-4 has one tardy job: job 4 with weight 1
Minimize Flowtime with no tardy jobs

- for all jobs to be on time, the last job must be on time

- schedulable set of jobs contain all jobs with due dates greater than or equal to the sum of all processing times

- Start from the end and choose the job with the largest proc time among the schedulable jobs, schedule this job last, remove from the list and continue

Optimal algorithm! (corresponding alg. For weighted flowtime is only heuristic)

Problem data
- Job i: 1, 2, 3, 4, 5
- \( p_i \): 4, 2, 3, 2, 4
- due date: 16, 11, 10, 9, 12
Operations Scheduling

- Step 1: Sum of the processing time is 15
  Job 1 has a due-date greater to 15 ⇒ schedule x-x-x-x-1

- Step 2: Sum of the remaining processing-times is 11
  Job 5 has a larger processing time ⇒ schedule x-x-x-5-1

- Step 3: remaining processing time is 7
  All remaining jobs have due dates at least that big
  ⇒ choose the one with the largest processing time ⇒ x-x-3-5-1

- Step 4: Continue ⇒ 2-4-3-5-1
Operations Scheduling

- **Minimizing total Tardiness**
  - General single-machine tardiness problem is NP-hard

- **Heuristic approach for the weighted problem (Rachamadugu/Morton)**
  - If all jobs are tardy, minimizing weighted tardiness is equivalent to minimizing weighted completion time, which is accomplished by the WSPT sequence.

- Weight-to-processing-time ratio is used

- Slack of job i, \( S_i = d_i - (p_i + t) \)  where \( t \) is the current time
A job should not get full WTPTR „credit“ if its slack is positive

\[ S_i^+ = \max \{0, S_i\} \]

Average processing time of the jobs:

\[ p_{av} = \frac{1}{n} \sum_{i=1}^{n} p_i \]

Ratio of the slack to the average processing time of jobs:

\[ \frac{S_i^+}{p_{av}} \]

which is the number of average job lengths until job j is tardy

Weight of a job is discounted by an exponential function:

\[ \exp\left(-\frac{S_i^+}{\kappa p_{av}}\right) \]
Operations Scheduling

Define the priority of job $i$ by

$$\gamma_i = \left( \frac{w_i}{p_i} \right) e^{-\frac{S_i^+}{\kappa \cdot p_{av}}}$$

$\kappa$ is a parameter of the heuristic to be chosen by the user (e.g. $\kappa = 2$)

Sequence jobs in descending order of priorities.
Operations Scheduling

Rachamadugu and Morton (1982) R&M Heuristics:

- The owner of Pensacola Boat Construction has currently 10 boats to construct;
- If PBC delivers a boat after the delivery date, a penalty proportional to both the value of the boat and the tardiness must be paid.

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(i)</td>
<td>8</td>
<td>12</td>
<td>6</td>
<td>10</td>
<td>3</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>7</td>
</tr>
<tr>
<td>w(i)</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>d(i)</td>
<td>26</td>
<td>28</td>
<td>32</td>
<td>35</td>
<td>38</td>
<td>48</td>
<td>50</td>
<td>51</td>
<td>53</td>
<td>64</td>
</tr>
</tbody>
</table>

- How should PBC schedule the work to minimize the penalty paid?
Penalty is weighted tardiness where weights measure the value of the boat.

κ = 2

Calculate: \( p_{av} = 9 \)

Job1:

\[
\gamma_1 = \left( \frac{W_1}{p_1} \right) e^{-\left[ S^+_{1}/(\kappa \ p_{av}) \right]} = \left( \frac{4}{8} \right) e^{-\left[ (26-8)/(2 \times 9) \right]} = 0,5 e^{-1} = 0,18
\]
## Operations Scheduling

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(i)</td>
<td>8</td>
<td>12</td>
<td>6</td>
<td>10</td>
<td>3</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>7</td>
</tr>
<tr>
<td>w(i)</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>d(i)</td>
<td>26</td>
<td>28</td>
<td>32</td>
<td>35</td>
<td>38</td>
<td>48</td>
<td>50</td>
<td>51</td>
<td>53</td>
<td>64</td>
</tr>
<tr>
<td>w(i)/p(i)</td>
<td>0,5</td>
<td>0,08</td>
<td>1</td>
<td>0,5</td>
<td>0,33</td>
<td>0,36</td>
<td>0,56</td>
<td>0,82</td>
<td>0,62</td>
<td>0,14</td>
</tr>
<tr>
<td>S(i)</td>
<td>18</td>
<td>16</td>
<td>26</td>
<td>25</td>
<td>35</td>
<td>37</td>
<td>41</td>
<td>40</td>
<td>40</td>
<td>57</td>
</tr>
<tr>
<td>S(i)/k av</td>
<td>1</td>
<td>0,89</td>
<td>1,44</td>
<td>1,39</td>
<td>1,94</td>
<td>2,06</td>
<td>2,28</td>
<td>2,22</td>
<td>2,22</td>
<td>3,17</td>
</tr>
<tr>
<td>priority</td>
<td>0,18</td>
<td>0,03</td>
<td>0,24</td>
<td>0,12</td>
<td>0,05</td>
<td>0,05</td>
<td>0,06</td>
<td>0,09</td>
<td>0,07</td>
<td>0,01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jobs</th>
<th>3</th>
<th>1</th>
<th>4</th>
<th>8</th>
<th>9</th>
<th>7</th>
<th>5</th>
<th>6</th>
<th>2</th>
<th>10</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>gamma_i</td>
<td>0,24</td>
<td>0,18</td>
<td>0,125</td>
<td>0,09</td>
<td>0,07</td>
<td>0,06</td>
<td>0,05</td>
<td>0,047</td>
<td>0,03</td>
<td>0,01</td>
<td></td>
</tr>
<tr>
<td>p_i</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>13</td>
<td>9</td>
<td>3</td>
<td>11</td>
<td>12</td>
<td>7</td>
<td>90</td>
</tr>
<tr>
<td>C_i</td>
<td>6</td>
<td>14</td>
<td>24</td>
<td>35</td>
<td>48</td>
<td>57</td>
<td>60</td>
<td>71</td>
<td>83</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>d_i</td>
<td>32</td>
<td>26</td>
<td>35</td>
<td>51</td>
<td>53</td>
<td>50</td>
<td>38</td>
<td>48</td>
<td>28</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>T_i</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>22</td>
<td>23</td>
<td>55</td>
<td>26</td>
<td>133</td>
</tr>
<tr>
<td>w_i</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>w_i T_i</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>35</td>
<td>22</td>
<td>92</td>
<td>55</td>
<td>26</td>
<td>230</td>
</tr>
</tbody>
</table>
Operations Scheduling

- Minimizing Earliness and Tardiness with a Common Due-Date

\[ Z = \sum_{i=1}^{n} (E_i + T_i) \]

- This is not a regular measure
- Assume common due date: \( d_j = D \)

- Number jobs in LPT sequence: \( p_1 \geq p_2 \geq \cdots \geq p_n \)
- Choose \( j^* = n/2 \) or \( n/2 + 0.5 \)

- If \( p_1 + p_3 + \cdots + p_{j^*} \leq D \) then the following sequence is optimal: 1 - 3 - 5 - 7 - \ldots - n - \ldots - 6 - 4 - 2
Operations Scheduling

Example: 10 Jobs with common due-date 80

<table>
<thead>
<tr>
<th>Jobs</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>proc Time</td>
<td>8</td>
<td>18</td>
<td>11</td>
<td>4</td>
<td>15</td>
<td>5</td>
<td>23</td>
<td>25</td>
<td>10</td>
<td>17</td>
</tr>
</tbody>
</table>
Operations Scheduling

- if \( p_1 + p_3 + \cdots + p_j > D \) then apply a heuristic (by Sundararaghavan & Ahmed, 1984)

- **Step 0:** Set \( B = D; A = \sum_{i=1}^{n} p_i - D; k = b = l; a = n; \) use the LPT sequence

- **Step 1:** If \( B > A \):
  - assign job \( k \) to position \( b \)
  - \( b := b + 1 \)
  - \( B := B - p_k \)
  - else
  - assign job \( k \) to position \( a \)
  - \( a := a - 1 \)
  - \( A := A - p_k \)

- **Step 2:** \( k := k - 1 \); if \( k \leq n \) go to step 1.
Operations Scheduling

Problems with non-zero release time

Non-zero release times typically makes scheduling problems much harder, e.g. SPT does in general not minimize total flowtime.

Heuristic Approach:
At each time $t$ determine the set of *schedulable jobs*: jobs that have been released but not yet processed.

Choose from the schedulable jobs according to some rule (e.g. SPT for minimizing flowtime)
## Operations Scheduling

*Preemption allowed:*

<table>
<thead>
<tr>
<th>j</th>
<th>r</th>
<th>p</th>
<th>t=0</th>
<th>t=2</th>
<th>t=3</th>
<th>t=4</th>
<th>t=6</th>
<th>t=9</th>
<th>t=10</th>
<th>t=11</th>
<th>t=12</th>
<th>t=18</th>
<th>t=24</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>8</td>
<td>rp 3</td>
<td>rp 4</td>
<td>rp 4</td>
<td>rp 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rp 2</td>
<td>rp 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Operations Scheduling

Minimizing makespan with non-zero release time and tails
Given $n$ jobs with release times $r_i$, processing times $p_i$, and tails $n_i$

Schrage Heuristics:
1. Start at $t=0$
2. Determine schedulable jobs
3. If there are schedulable jobs select the job $j^*$ among them with the largest tails, otherwise $t=t+1$ goto 1.
4. Schedule $j^*$ at $t$
5. If all jobs have been scheduled stop, otherwise set $t = t + p_{j^*}$, goto 1.
Operations Scheduling

- **Schrage Heuristics Example:** 6 jobs with release times and tails

<table>
<thead>
<tr>
<th>j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_j$</td>
<td>12</td>
<td>2</td>
<td>0</td>
<td>11</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>$p_j$</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$n_j$</td>
<td>21</td>
<td>9</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>

- **Minimize makespan!**
Operations Scheduling

- Denote by $S_J$ the set of schedulable jobs and by $S$ the scheduled sequence

- Step 1. $t = 0$, $S_J = \{3\}$, $S = <3>$, $t = 3$, $C_{\text{max}} = 5$
- Step 2. $t=3$, $S_J = \{2\}$, $S = <3-2>$, $t = 7$, $C_{\text{ma}} = 16$
- Step 3. $t = 9$, $S_J = \{5\}$, $S = <3-2-5>$, $t = 11$, $C_{\text{ma}} = 18$
- Step 4. $t=11$, $S_J = \{4, 6\}$, $S = <3-2-5-6>$, $t = 13$, $C_{\text{ma}} = 23$
- Step 5. $t=13$, $S_J = \{1, 4\}$, $S = <3-2-5-6-1>$, $t = 21$, $C_{\text{ma}} = 42$
- Step 6. $T=21$ $S_J = \{4\}$, $S = <3-2-5-6-1-4>$, $t = 27$, $C_{\text{ma}} = 42$

- Schrage heuristic is in general not optimal, e.g. B&B model can be used as an exact algorithm
Operations Scheduling

- **Minimizing Set-Up Times**
  - sequence-dependent set-up times
  - the time to change from one product to another may be significant and may depend on the previous part produced
  - \( p_{ij} = \text{time to process job } j \text{ if it immediately follows job } i \)

- Examples:
  - electronics industry
  - paint shops
  - injection molding

- minimizes makespan
- problem is equivalent to the traveling salesman problem (TSP), which is NP-hard.
Operations Scheduling

SST(=shortest set-up time) heuristic

A metal products manufacturer has contracted to ship metal braces each day to four customers. Each brace requires a different set-up on the rolling mill:

<table>
<thead>
<tr>
<th>Job</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>∞</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>∞</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>6</td>
<td>∞</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>4</td>
<td>∞*</td>
<td>∞</td>
</tr>
</tbody>
</table>

*Job C cannot follow job D, because of quality problems

SST-heuristic:

**Step 1** starting arbitrarily by choosing one Job: A

**Step 2** B has the smallest set-up time following A; ⇒ A-B

**Step 3** C has the smallest set-up time of all the remaining jobs following B; ⇒ A-B-C

**Step 4** D is the last remaining job; ⇒ A-B-C-D-A with a makespan of 3 + 4 + 2 + 5 = 14
Operations Scheduling

*z A regret based Algorithm
  ▶ makespan must be at least as big as the n smallest elements
  ▶ reduced matrix
    ▶ row reduction
    ▶ column reduction
    ▶ sum of reduced costs = lower bound for TSP
  ▶ find reduced matrix!

<table>
<thead>
<tr>
<th>Job</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>∞</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>∞</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>6</td>
<td>∞</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>4</td>
<td>∞*</td>
<td>∞</td>
</tr>
</tbody>
</table>
Operations Scheduling

- The reduced matrix has a zero in every row and column
- what happens if we do not choose j to follow i
- regret: lower bound on not choosing j to follow i

<table>
<thead>
<tr>
<th>Job</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>∞</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>∞</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>5</td>
<td>∞</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>∞*</td>
<td>∞</td>
</tr>
</tbody>
</table>
Regret heuristic

Find the cycle sequence that minimizes the set-up time.

Set-up times

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>18</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>∞</td>
<td>9</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>18</td>
<td>∞</td>
<td>13</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>∞</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>1</td>
<td>13</td>
<td>17</td>
<td>∞</td>
</tr>
</tbody>
</table>

Solution: TSP model – regret heuristic

Step 0  \( C(\text{max}) = 0 \) and \( L = 1 \)

Step 1  Reduce the matrix:

Reduced matrix

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>∞</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>9</td>
<td>∞</td>
<td>4</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>∞</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>0</td>
<td>12</td>
<td>16</td>
<td>∞</td>
<td>19</td>
</tr>
</tbody>
</table>

Step 1  Reduce the matrix:
Operations Scheduling

Step 2 Calculate the regret

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>15</td>
<td>0(0)</td>
<td>0(4)</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>∞</td>
<td>4</td>
<td>5</td>
<td>0(5)</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0(9)</td>
<td>9</td>
<td>∞</td>
<td>4</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
<td>0(1)</td>
<td>∞</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>0(17)</td>
<td>12</td>
<td>16</td>
<td>∞</td>
<td>1</td>
</tr>
</tbody>
</table>

Step 3 Choose the largest regret : 17

Step 4 Assign a job pair: Job 2 immediately follows job 5 (5-2)

L = 1+1;
We prohibit 2-5

New matrix

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>4</td>
<td>5</td>
<td>∞</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>∞</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0</td>
<td>∞</td>
<td>1</td>
</tr>
</tbody>
</table>
Operations Scheduling

**Step 1** Reduce the matrix

\[ C_{\text{max}} = 19 + 4 + 1 = 24 \]

Reduced Matrix

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>∞</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>∞</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

**Step 2** Calculate the regret

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>∞</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>∞</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

**Step 3** Choose the largest regret: 9

**Step 4** Assign a job pair: 3-1

Prohibit 1-3

**Step 1** Reduce the matrix: not possible

Matrix

<table>
<thead>
<tr>
<th>Job</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>∞</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>
Operations Scheduling

**Step 2** Calculate regret

<table>
<thead>
<tr>
<th>Job</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>0(3)</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0(1)</td>
<td>1</td>
<td>∞</td>
</tr>
<tr>
<td>4</td>
<td>0(0)</td>
<td>∞</td>
<td>0(2)</td>
</tr>
</tbody>
</table>

**Step 3** Choose the largest regret: 3

**Step 4** Assign job pair: 1-4; partial sequence: 5-2, 3-1-4
   - Prohibit 4-1 and 4-3 (to keep 3-1-4-3 from being chosen)

**Final Matrix**

<table>
<thead>
<tr>
<th>Job</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>∞</td>
</tr>
<tr>
<td>4</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

choose 2-3 and 4-5
-> sequence 3-1-4-5-2
the total set-up time is 24
Operations Scheduling

**Branch and Bound Algorithm**

1. Using the regret heuristic construct a (sub-)tree where each node represents the decision to let \( j \) follow \( i \) (\( i - j \)) or to prohibit that \( j \) follows \( i \) (\( i - j \)).

2. For each node a lower bound for the makespan is inferred from the regret heuristic.

3. Once a solution is obtained from the regret heuristic this is an upper bound for the optimal makespan. All nodes where the lower bound is above that level are pruned.

4. If all but one final node are pruned (or no non-pruned node can be further branched) this final node gives the optimal solution.

5. If 4. does not hold start again with 1. at one of nodes which are not pruned and can still be branched.
Operations Scheduling

- Branch and Bound Algorithm

All final nodes can be pruned:

opt. Solution has been found!
Operations Scheduling

- **Single-Machine Search Methods**
  - Neighborhood Search
  - Simulated Annealing
  - Ant System
  - Tabu Search
  - ...

- **Neighborhood Search**
  - seed
  - Neighborhood
  - any heuristic can be used to produce an initial sequence
Operations Scheduling

- adjacent pairwise interchange (API):
  - n-1 neighbors
  - 1-2-3-4-5-6-7-8-9
  - 1-2-3-4-6-5-7-8-9

- Pairwise interchange (PI):
  - n(n-1)/2 neighbors
  - 1-2-8-4-5-6-7-3-9

- Insertion (INS)
  - (n-1)^2 neighbors
  - 1-2-3-7-4-5-6-8-9

- Evaluation function
- Update function
Consider the following single-machine tardiness problem; Use the EDD sequence as the initial seed with an API neighborhood;

Data for neighborhood search

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing time</td>
<td>10</td>
<td>3</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Due-date</td>
<td>15</td>
<td>16</td>
<td>24</td>
<td>30</td>
<td>35</td>
<td>37</td>
</tr>
</tbody>
</table>

- Step 1: Construct the EDD sequence and evaluate its total tardiness
  - Set $i = 1$ and $j = 2$

- The EDD sequence $S^*$: 1-2-3-4-5-6; tardiness-vector (0, 0, 5, 7, 6, 14)
Operations Scheduling

- Step 2 Swap the jobs in the $i$-th and $j$-th position in $S^*$; the sequence is $S'$ with tardiness $T'$. If $T' < T$, go to step 4.

- Step 3 $j = j + 1$: If $j > n$: go to step 5. Otherwise, $i = j - 1$ and go to step 2;

- Step 4 Replace $S^*$ with $S'$; $i = 1$, $j = 2$; go to step 2;

- Step 5 Stop; $S^*$ is a local optimal sequence.
## Operations Scheduling

### Neighborhood search solution

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Schedule</th>
<th>Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>j</td>
<td>1 2 3 4 5 6</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2 1 3 4 5 6</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1 3 2 4 5 6</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1 2 4 3 5 6</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1 2 3 5 4 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Schedule</th>
<th>Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2 1 3 5 4 6</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1 3 2 5 4 6</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1 2 5 3 4 6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1 2 3 4 5 6</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>1 2 3 5 6 4</td>
</tr>
</tbody>
</table>
Operations Scheduling

**Single machine results**
- Flowtime - SPT (E)
- Lateness - SPT (E)
- Weighted Flowtime - WSPT (E)
- Maximal Tardiness (Lateness) - EDD (E)
- Nb. Of tardy jobs - Hodgson (E)
- weighted nb. Of tardy jobs - modified Hodgson (H)
- No jobs tardy/flowtime - modified SPT (E)
- Tardiness - R&M (H)
- weighted Tardiness - R&M (H)
- makespan with non-zero release time and tails - Schrage (H)
- Sequence dependent - SST (H), regret (H), B&B (E)
Operations Scheduling

**Parallel Machines**
- Scheduling decisions:
  - which machine processes the job
  - in what order

**List Schedule**
- to create a schedule, assign the job on the list to the machine with the smallest amount of work assigned.
- **Step 0.** Let $H_i=0$, $i=1,2,...,m$ be the assigned workload on machine $i$, $L=(\{1\},\{2\},...,\{n\})$ the ordered list sequence, $C_j=0$, $j=1,2,...,n$, and $k=1$
- **Step 1.** Let $j^* = L_k$ and $H_{i^*} = \min_{i=1,m} \{H_i\}$; Assign job $j^*$ to be processed on machine $i^*$, $C_{j^*} = H_{i^*} + p_{j^*}, H_{i^*} = H_{i^*} + p_{j^*}$
- **Step 2.** Set $k=k+1$, if $k>n$, stop. Otherwise go to step 1.
Operations Scheduling

Minimizing flowtime on parallel processors

Consider a facility with 3 identical machines and 15 jobs that need to be done as soon as possible;
Processing times (after SPT):

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>14</td>
<td>16</td>
<td>18</td>
<td>19</td>
</tr>
</tbody>
</table>

Optimal schedule:

<table>
<thead>
<tr>
<th>Machine 1</th>
<th>Machine 2</th>
<th>Machine 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>j</td>
<td>p(j)</td>
<td>C(j)</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>30</td>
</tr>
<tr>
<td>13</td>
<td>16</td>
<td>46</td>
</tr>
</tbody>
</table>

Total flowtime = 372
Operations Scheduling

Minimize the makespan

Use a longest processing time (LPT) first list;
Assign the next job on the list to the machine with the least total processing time assigned.

Optimal schedule:

<table>
<thead>
<tr>
<th>j</th>
<th>p(j)</th>
<th>C(j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>32</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>42</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>52</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>53</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>j</th>
<th>p(j)</th>
<th>C(j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>13</td>
<td>31</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>42</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>51</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>54</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>j</th>
<th>p(j)</th>
<th>C(j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td>30</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>42</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>48</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>52</td>
</tr>
</tbody>
</table>

![Diagram of optimal schedule]
Operations Scheduling

- **Flow shops**
  - all jobs are processed in the same order
  - two machine makespan model: Johnson’s Algorithm
  - Bound on makespan:
    
    \[
    C^* \geq \max \left\{ \min_{i=1,n} p_{i2} + \sum_{i=1}^{n} p_{i1}, \min_{i=1,n} p_{i1} + \sum_{i=1}^{n} p_{i2} \right\}
    \]

- Formulate Johnson’s Algorithm

- For 2-machine Flow shops the optimal schedule is a **Permutation Schedule**, i.e. the job sequence is the same on every machine
Operations Scheduling

- **Makespan with more than two machines**
  - Johnson’s algorithm will work in special cases, e.g. three machine problem where the second machine is dominated:
    \[ p_{i2} \leq \max(\min p_{i1}, \min p_{i3}) \]
  - Formulate an artificial two machine problem with
    \[ p'_{i1} = p_{i1} + p_{i2} \quad \text{and} \quad p'_{i2} = p_{i2} + p_{i3} \]
    and solve it using the Johnson algorithm gives the optimal solution for the three machine problem
Operations Scheduling

*Heuristics for the m-machine problem*

- Cambell, Dudek and Smith (1970)
- convert a m-machine problem into a two machine problem
- how?

\[ p'_{i1} = \sum_{j=1}^{k} p_{ij} \quad \text{and} \quad p'_{i2} = \sum_{j=l}^{m} p_{ij} \]

- Start with: \( k=1 \) and \( l=m \); then \( k=2 \) and \( l=m-1 \); until: \( k=m-1 \) and \( l=2 \)
- \( m-1 \) schedules are generated
- Use the best of these \( m-1 \) schedules
Operations Scheduling

Flow-shop heuristics

Processing data:

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>1</td>
<td>10</td>
<td>17</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>M2</td>
<td>13</td>
<td>12</td>
<td>9</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>M3</td>
<td>6</td>
<td>18</td>
<td>13</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>M4</td>
<td>2</td>
<td>18</td>
<td>4</td>
<td>6</td>
<td>16</td>
</tr>
</tbody>
</table>

Use the CDS heuristic to solve this problem.

(1) i.) Use the Johnson's algorithm only for M1 and M4:

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>1</td>
<td>10</td>
<td>17</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>M4</td>
<td>2</td>
<td>18</td>
<td>4</td>
<td>6</td>
<td>16</td>
</tr>
</tbody>
</table>

1-2-5-4-3, \( C_{\text{max}} = 88 \)

Next combine M1 and M2 to pseudomachine 1 and M3 and M4 to pseudomachine 2.

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM1</td>
<td>14</td>
<td>22</td>
<td>26</td>
<td>29</td>
<td>14</td>
</tr>
<tr>
<td>PM2</td>
<td>8</td>
<td>36</td>
<td>17</td>
<td>8</td>
<td>21</td>
</tr>
</tbody>
</table>

[\( j \)] 1 2 3 4 5

5-2-3-1-4, \( C_{\text{max}} = 85 \)

Finally combine M1, M2 and M3 to pseudomachine 1 and M2, M3 and M4 to pseudomachine 2.

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM1</td>
<td>20</td>
<td>40</td>
<td>39</td>
<td>31</td>
<td>19</td>
</tr>
<tr>
<td>PM2</td>
<td>21</td>
<td>48</td>
<td>26</td>
<td>25</td>
<td>24</td>
</tr>
</tbody>
</table>

[\( j \)] 1 2 3 4 5

5-1-2-3-4, \( C_{\text{max}} = 85 \)
## Operations Scheduling

- **Gantt Chart for the CDS schedule**

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

- Time: 0 10 20 30 40 50 60 70 80
Operations Scheduling

**Gupta - Heuristic**

- Gupta (1972)
- Exact for 2-machine problem and 3-machine problem, where the 2nd machine is dominated

\[
e_i = \begin{cases} 
1 & \text{if } p_{i1} < p_{im} \\
-1 & \text{if } p_{i1} \geq p_{im}
\end{cases}
\]

\[
s_i = \min_{k=1, \ldots, m-1} \left\{ \frac{e_i}{p_{i,k} + p_{i,k+1}} \right\}
\]

- Sorting jobs with nonincreasing \( s_i \)

\((s_{[1]} \geq s_{[2]} \geq \ldots \geq s_{[n]})\)

<table>
<thead>
<tr>
<th>Job</th>
<th>p1+p2</th>
<th>p2+p3</th>
<th>p3+p4</th>
<th>min</th>
<th>ei</th>
<th>si</th>
<th>[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
<td>19</td>
<td>8</td>
<td>8</td>
<td>1</td>
<td>0.12</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>22</td>
<td>30</td>
<td>36</td>
<td>22</td>
<td>1</td>
<td>0.05</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>22</td>
<td>17</td>
<td>17</td>
<td>-1</td>
<td>-0.06</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>29</td>
<td>19</td>
<td>8</td>
<td>8</td>
<td>-1</td>
<td>-0.12</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>8</td>
<td>21</td>
<td>8</td>
<td>1</td>
<td>0.12</td>
<td>2</td>
</tr>
</tbody>
</table>
Operations Scheduling

- **Branch and Bound Approaches**
  - machine based bounds
  - job based bounds
  - three machines
    - $H_j =$ current completion time of the last job scheduled on machine $j$
    - $U =$ set of unscheduled jobs
    - makespan on machine 1 must be at least:
      \[
      C^*_{\text{max}} \geq H_1 + \sum_{i \in U} p_{i1} + \min_{i \in U} \{p_{i2} + p_{i3}\}
      \]
      - machine 2:
      \[
      C^*_{\text{max}} \geq \max \left\{\left[H_1 + \min_{i \in U} \{p_{i1}\}\right] H_2 + \sum_{i \in U} p_{i2} + \min_{i \in U} \{p_{i3}\}\right\}
      \]
Operations Scheduling

Machine 3:

\[ C^*_\text{max} \geq \max \left\{ \left( H_1 + \min_{i \in U} \{ p_{i1} + p_{i2} \} \right), \left[ H_2 + \min_{i \in U} \{ p_{i2} \} \right], H_3 \right\} + \sum_{i \in U} p_{i3} \]

Job oriented bounds:

\[ C_{\text{max}} \geq H_1 + \max_{i \in U} \left\{ \sum_{j=1}^{m} p_{ij} + \sum_{k \in U, k \neq i} \min \{ p_{k1}, p_{k3} \} \right\} \]
Operations Scheduling

- **B&B algorithm for minimizing makespan in multi-machine Flow Shops**

  1. Create an initial incumbent solution, e.g. CDS heuristic
     - upper bound
  2. Starting at t=0 with a root node; branch the tree by generating a node for each schedulable jobs.
  3. In each node calculate the lower bounds and prune the node if at least one exceeds the upper bound.
  4. If a non-pruned final node exists at the lowest level take the corresponding solution as new incumbent, update the upper bound and do the corresponding pruning.
  5. If all final nodes are pruned current incumbent is the optimal solution, otherwise branch at the node with the lowest lower bound and goto 3.
Operations Scheduling

Makespan permutation schedule for a three-machine flow-shop

Processing data:

<table>
<thead>
<tr>
<th>Machine</th>
<th>Job 1</th>
<th>Job 2</th>
<th>Job 3</th>
<th>Job 4</th>
<th>Job 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>17</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>12</td>
<td>9</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>18</td>
<td>13</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

Solution:
Start with CDS algorithm: sequence: 1-2-3-4-5, C_{max} = 65

Initial lower bound:
M1: C_{max}^{*} \geq H_1 + (p_{11} + p_{21} + p_{31} + p_{41} + p_{51})
  + \min\{p_{12} + p_{13}, p_{22} + p_{23}, p_{32} + p_{33}, p_{42} + p_{43}, p_{52} + p_{53}\}
  = 0 + (1 + 10 + 17 + 12 + 11) + \min\{19, 30, 22, 19, 8\} = 51 + 8 = 59

M2: C_{max}^{*} \geq \max\{H_1 + \min\{p_{11}, p_{21}, p_{31}, p_{41}, p_{51}\}, H_2\}
  + (p_{12} + p_{22} + p_{32} + p_{42} + p_{52}) + \min\{p_{13}, p_{23}, p_{33}, p_{43}, p_{53}\}
  = \max\{[0 + \min\{1, 10, 17, 12, 11\}, 0\}
  + (13 + 12 + 9 + 17 + 3) + \min\{6, 18, 13, 2, 5\}
  = 1 + 54 + 2 = 57
Operations Scheduling

M3: $C_{\text{max}}^* \geq \max\{H_1 + \min\{p_{11} + p_{12}, p_{21} + p_{22}, p_{31} + p_{32}, p_{41} + p_{42}, p_{51} + p_{52}\}, \]
$[H_2 + \min\{p_{12}, p_{22}, p_{32}, p_{42}, p_{52}\}], H_3\} + (p_{13} + p_{23} + p_{33} + p_{43} + p_{53})$
$=\max\{0 + \min\{14, 22, 26, 29, 14\}, \]
$[0 + \min\{13, 12, 9, 17, 3\}], 0\} + (6 + 18 + 13 + 2 + 5)$
$=\max\{14, 3, 0\} + 44 = 58$

Job-based bounds are the following:

$C_{\text{max}} \geq H_1 + \sum_{j=1}^{3} p_{1j} + \sum_{k \in \{2, 3, 4, 5\}} \min\{p_{k1}, p_{k3}\}$

J1: $C_{\text{max}}^* \geq H_1 + (p_{11} + p_{12} + p_{13})$
$+(\min\{p_{21}, p_{23}\} + \min\{p_{31}, p_{33}\} + \min\{p_{41}, p_{43}\} + \min\{p_{51}, p_{53}\})$
$=0 + (1 + 13 + 6) + (\min\{10, 18\} + \min\{17, 13\} + \min\{12, 2\} + \min\{11, 5\})$
$=0 + 20 + (10 + 13 + 2 + 5) = 50$

Similarly, we have

J2: $C_{\text{max}}^* \geq 61$, J3: $C_{\text{max}}^* \geq 57$, J4: $C_{\text{max}}^* \geq 60$, J5: $C_{\text{max}}^* \geq 45$

LB: 61, UB: 65
Production Management 231

UB = 65 (Gupta)
LB = 61 J2

Solution (=LB): 65
Operations Scheduling

1st level: J2 at first place: $H_1 = 10$, $H_2 = 22$, $H_3 = 40$
   $U = \{1, 3, 4, 5\}$
   M1: $C_{\text{max}}^{\ast} \geq 59$
   M2: $C_{\text{max}}^{\ast} \geq 66$, which is greater than the upper bound; thus we fathom the node;
J3, J4 and J5 at first place: we can fathom all of them;
2nd level: Consider Job 3: $H_1 = 18$, $H_2 = 27$, $H_3 = 40$, $U = \{2, 4, 5\}$
   M1: $C_{\text{max}}^{\ast} \geq 59$
   M2: $C_{\text{max}}^{\ast} \geq 62$
   M3: $C_{\text{max}}^{\ast} \geq 65$, so we fathom the job; only job 2 remains unfathomed;
3rd level: Job 3: $H_1 = 28$, $H_2 = 37$, $H_3 = 57$, $U = \{4, 5\}$
   M1: $C_{\text{max}}^{\ast} \geq 59$
   M2: $C_{\text{max}}^{\ast} \geq 61$
   M3: $C_{\text{max}}^{\ast} \geq 64$
   Machine-bounds did not fathom the node; so we have to calculate job-based bounds:
   J4: $C_{\text{max}}^{\ast} \geq 64$
   J5: $C_{\text{max}}^{\ast} \geq 49$

$\Rightarrow$ best bound = 64; thus create nodes for J4 and J5

4th level: nodes J4 and J5 of level 3 will be fathomed; thus the algorithm is complete:
1-2-3-4-5 with a makespan of 65;
Operations Scheduling

- Job Shops
  - different routings for different jobs
  - precedence constraints
  - \((n!)^m\) possible schedules
Operations Scheduling

- Two machine job shops
  - Jackson (1956)
  - minimize makespan
    - Machine A: \{AB\}, \{A\}, \{BA\}
    - Machine B: \{BA\}, \{B\}, \{A,B\}

Jackson’s algorithm

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route</td>
<td>BA</td>
<td>AB</td>
<td>BA</td>
<td>B</td>
<td>A</td>
<td>AB</td>
<td>B</td>
<td>BA</td>
<td>BA</td>
<td>AB</td>
</tr>
<tr>
<td>p(i)1</td>
<td>3</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>3</td>
<td>9</td>
<td>0</td>
<td>8</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>p(i)2</td>
<td>8</td>
<td>10</td>
<td>13</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

Find a schedule that would finish all jobs as soon as possible!

Solution:
\{A\} = \{5\}, \{B\} = \{4,7\}, \{AB\} = \{2, 6, 10\} and \{BA\} = \{1, 3, 8, 9\}
Operations Scheduling

Johnson’s algorithm for \{AB\}:

<table>
<thead>
<tr>
<th>Job</th>
<th>2</th>
<th>10</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(i)1</td>
<td>1</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>p(i)2</td>
<td>10</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

Johnson’s algorithm (reversed) for \{BA\}:

<table>
<thead>
<tr>
<th>Job</th>
<th>9</th>
<th>3</th>
<th>8</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(i)1</td>
<td>13</td>
<td>11</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>p(i)2</td>
<td>6</td>
<td>13</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

sequence for A: 2-10-6-5-9-3-8-1
sequence for B: 9-3-8-1-4-7-2-10-6

makespan: 67
Operations Scheduling

**Dispatching**

- job shop scheduling
- dispatching rules
- Basic idea:
  - schedule an operation of a job as soon as possible
  - if more than one job is waiting to be processed by the same machine, schedule the one with best priority

Define:

- $A =$ set of idle machines
- $J_k =$ the index of the last job scheduled on machine $k$
- $U_k =$ the set of jobs that can be processed on machine $k$
- $H_k =$ the completion time of the job currently processed on machine $k$
- $u_{it} =$ the priority of job $i$ at time $t$
Operations Scheduling

Step 0. Initialize: \( t=0; H_k=0, k=1,2,...,m; \)
\( A=\{1,2,...,m\}; U_k=\{i|\text{operation 1 of } i \text{ is on machine } k, \)
\( i=1,2,...,n\}; s_{ij}=c_{ij}=0. \) Go to step 4.

Step 1. Increment \( t; \)

Let
\[ t = \min_{k=1, m; k \notin A} H_k, \text{ and } K = \{k \mid H_k = t\} \]

Step 2. Find the job or jobs that complete at time \( t \) and the machine released. Set \( A = A \cup K. \)

Step 3. Determine the jobs ready to be scheduled on each machine;
Let \( U_k=\{i|\text{job } i \text{ uses machine } k \text{ and all operations of job } i \)
\( \text{before machine } k \text{ are completed}\}, k=1,2,...,m. \)
If \( U_k=0 \text{ for } k=1,2,...,m, \text{Stop.} \)
If \( U_k=0 \text{ for } k \in A, \) go to Step 1.
Operations Scheduling

Step 4. For each idle machine try to schedule a job;
for each \( k \in A \) with \( U_k \neq 0 \),

let \( i^* \) be the job with the best priority:
\[
\min_{i \in U_k} u_{i*t} = \min_{i \in U_k} u_{it}
\]

Schedule job \( i^* \) on machine \( k \)
Set \( J_k = i^, s_{i^*k} = t, c_{i^*k} = t + p_{i^*(k)}, H_k = c_{i^*k} \)

Remove the scheduled job from \( U_k \)
\( U_k \leftarrow U_k - \{i\} \)
and the machine from \( A \)
\( A \leftarrow A - \{k\} \)
Go to Step 1
Operations Scheduling

Many priority measures possible:

- SPT
- FCFS
- MWKR (most work remaining)
- EDD
- EDD/OP
- SLACK, SLACK/OP
- Critical ratio: slack/remaining time
- ...
Quick Closures: job-shop dispatch heuristic

Quick Closure has four machines in the shop: (1) brake, (2) emboss, (3) drill, (4) mill; The shop has currently orders for six different parts, which use all the four machines, but in a different order.

Processing time:

<table>
<thead>
<tr>
<th>Job</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6/1 8/2 13/3 5/4</td>
</tr>
<tr>
<td>2</td>
<td>4/1 1/2 4/3 3/4</td>
</tr>
<tr>
<td>3</td>
<td>3/4 8/2 6/1 4/3</td>
</tr>
<tr>
<td>4</td>
<td>5/2 10/1 15/3 4/4</td>
</tr>
<tr>
<td>5</td>
<td>3/1 4/2 6/4 4/3</td>
</tr>
<tr>
<td>6</td>
<td>4/3 2/1 4/2 5/4</td>
</tr>
</tbody>
</table>

Finish all six parts as soon as possible!

Solution: We use a dispatch procedure with MWKR as the priority.
Operations Scheduling

Step 1 \( t = 0, H_1 = H_2 = H_3 = H_4 = 0, A = \{1, 2, 3, 4\}, U_1 = \{1, 2, 5\}, U_2 = \{4\}, U_3 = \{6\}, U_4 = \{3\}; s_{ij} = c_{ij} = 0, i = 1, 2, 3, 4, 5, 6; \) and \( j = 1, 2, 3, 4; \) Go to step 4

Step 4 \( u_{10} = -(6+8+13+5) = -32, u_{20} = -12, u_{50} = -17; \) thus \( s_{11} = 0, c_{11} = 0 + 6 = 6, H_1 = 6. \)
Remove job 1 from \( U_1, U_1 = \{2, 5\} \) and machine 1 from \( A, A = \{2, 3, 4\}. \)
Set \( k = 2; \) there is only one job in \( U_2 \) so we schedule it on machine 2; \( i^* = 4, s_{41} = 0, c_{41} = 5, H_2 = 5, U_2 = \{\}, \) and \( A = \{3, 4\}. \)
We schedule J6 and J3 on M3 and M4 (tab: \( t = 0 \) row). Go to step 1.

Step 1 \( t = \min_{k=1,m:k \in A} H_k = \min\{6, 5, 4, 3\} = 3, \) and \( K = \{k \mid H_k = 3\} = \{4\}; H_k \text{ min is bold in the table}; \)

Step 2 J3 completes at time 3 on M4, so \( i^3 = \{i \mid J_k = i, k \in K\} = \{3\}, K = \{4\}, \) and \( A = \{\} \cup \{4\} = \{4\}, \) (tab: \( t = 3 \) row)

Step 3 \( U_1 = \{2, 5\}, U_2 = \{3\}, U_3 = U_4 = \{\}; \) Since no jobs are waiting for M4, no jobs can be scheduled to start at time 3; go to step 1 etc.
### Operations Scheduling

<table>
<thead>
<tr>
<th>t</th>
<th>it</th>
<th>K</th>
<th>A</th>
<th>U1</th>
<th>U2</th>
<th>U3</th>
<th>U4</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
<th>H4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>-</td>
<td>1,2,3,4</td>
<td>1,2,5</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2,5</td>
<td>3</td>
<td></td>
<td></td>
<td>6</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>3</td>
<td>3,4</td>
<td>2,5,6</td>
<td>3</td>
<td></td>
<td></td>
<td>6</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2,3,4</td>
<td>2,4,5,6</td>
<td>3</td>
<td></td>
<td></td>
<td>6</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1,3,4</td>
<td>2,4,5,6</td>
<td>1</td>
<td>16</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>2</td>
<td>2,3,4</td>
<td>2,3,5,6</td>
<td>1</td>
<td>16</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>1</td>
<td>1,3,4</td>
<td>2,3,5,6</td>
<td>4</td>
<td>19</td>
<td>21</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>5</td>
<td>1</td>
<td>1,4</td>
<td>2,3,6</td>
<td>5</td>
<td></td>
<td></td>
<td>23</td>
<td>21</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>2</td>
<td>2,4</td>
<td>3,6</td>
<td>5</td>
<td>1</td>
<td>23</td>
<td>25</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>1</td>
<td>1,4</td>
<td>3,6</td>
<td>2</td>
<td>1</td>
<td>25</td>
<td>25</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>6,5</td>
<td>1,2</td>
<td>1,2,4</td>
<td>3</td>
<td>2,6</td>
<td>1</td>
<td>5</td>
<td>31</td>
<td>29</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>29</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>31</td>
<td>30</td>
<td>31</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td>1,2</td>
<td>6</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>3,4</td>
<td>1,3</td>
<td>1,2,3</td>
<td></td>
<td></td>
<td>1,2,3</td>
<td>4,6</td>
<td>44</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>6</td>
<td>4</td>
<td>1,2,4</td>
<td></td>
<td></td>
<td>2,3,5</td>
<td>4</td>
<td>44</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>4</td>
<td>4</td>
<td>1,2,4</td>
<td></td>
<td></td>
<td>2,3,5</td>
<td></td>
<td>44</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>1</td>
<td>3</td>
<td>1,2,3,4</td>
<td></td>
<td></td>
<td>2,3,5</td>
<td>1</td>
<td>48</td>
<td>49</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>2</td>
<td>3</td>
<td>1,2,3</td>
<td>3,5</td>
<td>2</td>
<td>52</td>
<td>49</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>1</td>
<td>4</td>
<td>1,2,4</td>
<td>5</td>
<td>2</td>
<td>52</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>3,2</td>
<td>3,4</td>
<td>1,2,3,4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>5</td>
<td>3</td>
<td>1,2,3,4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Production Management 242
Operations Scheduling
Chapter 10

Section 5.5: Bottleneck Scheduling
Operations Scheduling

Shifting Bottleneck Procedure

heuristic to minimize makespan for multiple machine job shops

Main idea:
1. for each job on each machine calculate the minimal amount of time needed before and after the processing of this job
   generates minimal makespan problem with release times and tails
2. for each machine solve this problem for each machine (e.g. Schrage heuristic) and determine the machine with the maximal makespan (bottleneck machine)
3. Fix the found sequence on the bottleneck machine, update release times and tails on the remaining machines and repeat 2. for the remaining machines until schedules for all machines have been determined
Operations Scheduling

- **Shifting Bottleneck Procedure Example:**
  - 3 machines (M1, M2, M3), 3 jobs (1, 2, 3)

- **Job routings:**
  1: M1-M2-M3
  2: M2-M3-M1
  3: M2-M1-M3

- **Processing times:**

<table>
<thead>
<tr>
<th>$p_{ik}$</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
Operations Scheduling

Machine-Flow-Graph:

Job 1

Job 2

Job 3
**Operations Scheduling**

- **Problems with release times and tails for each machine:**

<table>
<thead>
<tr>
<th></th>
<th>M1:</th>
<th></th>
<th>M2:</th>
<th></th>
<th>M3:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>( r_j )</td>
<td>0</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>( p_j )</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>( n_j )</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

|    |    |    |    |    |    |
|----|----|----|----|----|
| \( r_j \) | 6   | 2  | 7   |    |    |
| \( p_j \)  | 2   | 3  | 1   |    |    |
| \( n_j \)  | 0   | 3  | 0   |    |    |
Operations Scheduling

Schrage heuristic gives the following solutions for the three machines:

Machine 2 is bottleneck with $C_2 = 11$

Fix sequence on machine 2
Operations Scheduling

- **Machine-flow-graph:**

![Graph Diagram]

- **Update release times and tails on M1 and M3:**

  **M1:**
  
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_j$</td>
<td>0</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>$p_j$</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$n_j$</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

  **M3:**
  
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_j$</td>
<td>9</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>$p_j$</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>$n_j$</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
Operations Scheduling

- **Schrage heuristic for M1, M3:**

  ![Diagram of scheduling operations]

  - Both machines could be considered the bottleneck with $C=12$, fix sequence on M1
Operations Scheduling

- Updated machine-flow-graph:

- update relase time and tails and apply Schrage to M3. This gives $C_{\text{max}} = 12$

- with $C_{\text{max}} = 12$
Operations Scheduling

Finite Capacity Scheduling

- MRP systems generally assume constant lead times, ignore setups
- MRP plans might be unrealistic
- Traditionally problem hidden by inventory and excess capacity
- Reducing inventory and capacity makes finite capacity scheduling crucial
- Computer-assisted finite capacity scheduling systems rather than manual scheduling by foreman
Operations Scheduling

- Work to do: 8.3abcde, 8.4, 8.5, 8.6, 8.10, 8.14, 8.16, 8.18 (with the following due dates: 42, 50, 12, 63, 23, 34, 36, 42, 54, 32) 8.30ab, 8.32abc, 8.36ab, 8.43, 8.44, 8.49ab, 8.51ab, 8.56, 8.57 (apply shifting bottleneck procedure)

- Minicase: Ilana Designs