Flow Shop Scheduling with Synchronous Material Movement

Kwei-Long Huang

Dept of Information Management
National Taiwan University

Information Management Seminar, National Taiwan University, Dec 12, 2008
Agenda

- Introduction
- T-line machining center with one machine
  - Complexity
  - Common unloading times
  - Dynamic programming algorithm
  - Lower bound
  - Heuristic algorithms and computational results
- T-line machining center with $m$ machines
  - Dynamic programming algorithm
  - Lower bound
- Conclusions and future research
Synchronous Material Movement

- An automated manufacturing cell which consists of a loading/unloading (L/U) station, $m$ machines, and a rotary table.
- The L/U station and these machines surround the rotary table.
Flow Shop Scheduling with Synchronous Material Movement
Flow Shop Scheduling with Synchronous Material Movement (con’t)

- An application to this type of manufacturing cells is called T-line machining center produced by Milacron Cincinnati
Agenda

- Introduction
- **T-line machining center with one machine**
  - Complexity
  - Common unloading times
  - Dynamic programming algorithm
  - Lower bound
  - Heuristic algorithms and computational results
- **T-line machining center with $m$ machines**
  - Dynamic programming algorithm
  - Lower bound
- Conclusions and future research
T-line Machining Center with One Machine

Problem assumptions:

- Given $n$ jobs and all jobs are available at time zero
- No preemption
- The machine can only process one job at a time
- The rotary table with two pallets
- Loading time, processing time, and unloading time for job $j$ denoted as $l_j$, $p_j$ and $u_j$, respectively
- Objective is to minimize the makespan ($C_{\text{max}}$)
  - Transportation times are neglected
Gantt Chart
Complexity: NP-hard

- The scheduling problem with the makespan objective in a T-line machining center is NP-hard
- Show this problem is equivalent to a problem known as NP-hard
- Numerical matching problem with target sums (NMTS)
  - Strongly NP-Complete (Garey and Johnson 1979)
  - Two sets \( A = \{a_1, a_2, \ldots, a_n\} \) and \( B = \{b_1, b_2, \ldots, b_n\} \)
  - Target set \( C = \{c_1, c_2, \ldots, c_n\} \)

Can \( A \cup B \) be partitioned into \( n \) disjoint sets \( D_k \), each containing exactly one element from set \( A \) and one element from set \( B \), such that,

\[
c_k = a_{[k]} + b_{[k]}, \quad a_{[k]}, b_{[k]} \in D_k \quad \text{for} \quad k = 1, \ldots, n
\]
Decision problem (P)

- Given a number of jobs with loading, processing and unloading times that have to be processed in a one-machine T-line machining center, does there exist a schedule, say \( \sigma \), with makespan \( C_{\text{max}}(\sigma) \) which is less than or equal to a given value \( Z \)?
Complexity (con’t)

- Given any instance of NMTS, the instance P1 for Problem P can be constructed as follows:
  - Contain three types of jobs
  - Ja-Type: \( l(Ja_i) = 1, \ p(Ja_i) = 3, \ u(Ja_i) = 2\beta + 3a_i, \ 1 \leq i \leq n, \)
  - Jb-Type: \( l(Jb_i) = \beta + 3b_i, \ p(Jb_i) = \beta + 1, \ u(Jb_i) = 1, \ 1 \leq i \leq n, \)
  - Jc-Type: \( l(Jc_i) = 2, \ p(Jc_i) = 3\beta + 3c_i, \ u(Jc_i) = \beta, \ 1 \leq i \leq n, \)
  - Z value: \( Z = 2 + 4n + 4n\beta + 3C_0 \)
    - where \( \beta = 3 \max\{c_i | 1 \leq i \leq n\} \) and \( C_0 = \sum_{i=1}^{n} c_i \)
Complexity (con’t) “the If part”

- Assume there exists a solution to NMTS
- A sequence can be formed as follows:

\[
C_{max} = 2 + 3n + 3n\beta + 3\sum_{i=1}^{n} c_i + n\beta + n = 2 + 4n + 4n\beta + 3C_0 = Z
\]
Complexity (con’t)  
“the Only If part”

- Assume that for instance $P1$ there exists a sequence, say $\sigma$, and its makespan $\leq Z$. Then we have to show that there exists a solution to NMTS
- What does sequence $\sigma$ look like?
- **Lemma 1**: on the CNC machine, the first and last cycles have one unit of idle time, respectively
- **Lemma 2**: on the L/U station, the second and second last cycles have one unit of idle time, respectively
Complexity (con’t)
“the Only If part”

- Lemma 3: Jobs in sequence $\sigma$ are sequenced in the following form
  $[Ja, Jc, Jb, Ja, Jc, Jb, \ldots, Ja, Jc, Jb]$

- Ideas of the proof
  - Consider first three jobs
  - If the first job is not Ja-type
  - If the second job is not Jc-type
  - If the third job is not Jb-type
Complexity (con’t) “the Only If part”

- If the first job is not Ja-type
  - The first job is Jb-type
    - L/U station
    - CNC
    - $\beta + 3b_i$
    - $\beta + 1$
    - idle time is $\beta + 3b_i$

- The first job is Jc-type
  - L/U station
  - CNC
  - 2
  - $3/\pi + 3c_i$
  - idle time is 2
Complexity (con’t)
“the Only If part”

- If the second job is not Jc-type
  - The second job is Ja-type

```
<table>
<thead>
<tr>
<th>L/U station</th>
<th>CNC</th>
<th>idle time is 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2(β+3a_i)</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
```

- The second job is Jb-type

```
<table>
<thead>
<tr>
<th>L/U station</th>
<th>CNC</th>
<th>idle time is (β+3b_i) -3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>β+1</td>
</tr>
<tr>
<td>β+3b_i</td>
<td>2(β+3a_i)</td>
<td></td>
</tr>
</tbody>
</table>
```

16
Complexity (con’t) “the Only If part”

- If the third job is not Jb-type
  - The third job is Ja-type
    - idle time is $\beta - 3a_{[1]} + 3c_{[1]} - 1$

<table>
<thead>
<tr>
<th>L/U station</th>
<th>1</th>
<th>2</th>
<th align="left">$2/+3a_{[1]}</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNC</td>
<td>3</td>
<td></td>
<td align="left">$3/+3c_{[1]}$</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

- The third job is Jc-type

<table>
<thead>
<tr>
<th>L/U station</th>
<th>1</th>
<th>2</th>
<th align="left">$2/+3a_{[1]}</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNC</td>
<td>3</td>
<td></td>
<td align="left">$3/+3c_{[1]}$</td>
<td></td>
<td>$3/+3c_{[1]}$</td>
</tr>
</tbody>
</table>
Complexity (con’t) “the Only If part”

- Lemma 4: In sequence $\sigma$, every three jobs are grouped as a block, we will have

$$a_{[k]} + b_{[k]} = c_{[k]}, \ k = 1, \ldots, n$$

- Ideas of the proof
  - Consider the first three jobs
  - If $a_{[1]} + b_{[1]} < c_{[1]}$
  - If $a_{[1]} + b_{[1]} > c_{[1]}$
Complexity (con’t)  
“the Only If part”

- If $a_{[1]} + b_{[1]} < c_{[1]}$

  ![Diagram showing the calculations for $a_{[1]} + b_{[1]} < c_{[1]}$]

- If $a_{[1]} + b_{[1]} > c_{[1]}$

  ![Diagram showing the calculations for $a_{[1]} + b_{[1]} > c_{[1]}$]
Complexity (con’t)
“the Only If part”

- Through Lemmas 1~4, we show that there exists a solution to NMTS
- Theorem 1: the scheduling problem with the makespan objective in a T-line machining center is NP-hard
Common Unloading Times

- Assume all unloading times are equal to a constant (denoted as c)
- Given a sequence, the schedule on machines looks like as follows

```
L/U station
<table>
<thead>
<tr>
<th>1st cycle</th>
<th>2nd cycle</th>
<th>3rd cycle</th>
<th>4th cycle</th>
<th>(n+1)th cycle</th>
</tr>
</thead>
</table>

CNC
```

Z
Common Unloading Times (con’t)

- Ignore the first two cycles and the last cycle, it will become a two-machine flow shop problem with blocking

- The Gilmore-Gomory (G-G) algorithm (1964) can solve the problem with blocking optimally
Given the first two jobs, say a and b, in an optimal sequence, the rest of job sequence can be generated by the G-G algorithm.

**Problem G:**

- Let $l_j = l_j + c$ ; $j \neq a$ and $b$
- Additional two jobs
  - $J_0$: $l_0 = 0$, $p_0 = p_b$
  - $J_{n+1}$: $l_{n+1} = c$ and $p_{n+1} = 0$
Common Unloading Times (con’t)

- **Lemma 5**: $J_0$ is sequenced in the first position
- **Lemma 6**: $J_{n+1}$ is sequenced in the last position
- The optimal makespan of Problem G is $Z^G$
- The optimal makespan of the original problem ($Z$) is $Z^G + l_a + \max(l_b, p_a) + c$

<table>
<thead>
<tr>
<th>L/U station</th>
<th>CNC</th>
<th>A time period</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_a$</td>
<td>$p_a$</td>
<td>(Z^G)</td>
</tr>
<tr>
<td>$l_b$</td>
<td>$p_b$</td>
<td></td>
</tr>
<tr>
<td>$l_3$</td>
<td>$p_{[3]}$</td>
<td></td>
</tr>
<tr>
<td>$l_4$</td>
<td>$c$</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>$c$</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>$p_{[n]}$</td>
<td></td>
</tr>
</tbody>
</table>
Common Unloading Times (con’t)

- Given any two jobs in the first two positions, apply the G-G algorithm to obtain the minimum makespan corresponding to that pair of jobs.
- The one with the smallest makespan among these $n(n-1)$ pairs of jobs forms an optimal sequence for the original problem.
- The complexity of the G-G algorithm is $O(n \log n)$.
- The complexity of the proposed algorithm is $O(n^3 \log n)$. 
DP Algorithm (One)

Optimal value function (OVF): \( f_i(S, g, j) \) = minimum completion time for processing job \( g \) on CNC, unloading the last job in \( S \) and loading job \( j \) at the L/U station, given that the \( i \) jobs in \( S \) have already been completed. 

\((4.4)\)
Recurrence relation (RR):

\[ f_i(S, g, j) = \min_{k \in S} \{ f_{i-1}(S \setminus \{k\}, k, g) + \max\{p_g, u_k + l_j\} \}; \quad i = 1, \ldots, n-2 \quad ; \quad \{g, j\} \subseteq N \quad ; \]

\[ S \subseteq N \setminus \{g, j\}, \quad |S| = i. \quad (4.6) \]

Boundary condition (BC): \[ f_0(\emptyset, g, j) = l_g + \max\{p_g, l_j\}; \quad \{g, j\} \subseteq N. \quad (4.7) \]

Answer (ANS):

\[ \min_{g \subseteq N} \{ f_{n-1}(S, g, \emptyset) \} \quad (4.8) \]

where \[ f_{n-1}(S, g, \emptyset) = \min_{k \in S} \{ f_{n-2}(S \setminus \{k\}, k, g) + \max\{p_g, u_k\} \} + u_g \quad ; \quad g \subseteq N \quad ; \quad S = N \setminus \{g\} \quad , \]

\[ |S| = n - 1. \quad (4.9) \]
## DP Algorithm (One) - Computational Analysis

<table>
<thead>
<tr>
<th>Stage</th>
<th>Number of combinations</th>
<th>Addition</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary condition (i = 0)</td>
<td>n(n-1)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Recurrence relation (1 ≤ i ≤ n−2)</td>
<td>n(n−1)C_i^{n−2}</td>
<td>2i</td>
<td>i + (i − 1)</td>
</tr>
<tr>
<td>Answer f_{n-1} (i = n−1)</td>
<td>n</td>
<td>n</td>
<td>n - 1 + n - 2</td>
</tr>
<tr>
<td>Minimum makespan</td>
<td>1</td>
<td>0</td>
<td>n - 1</td>
</tr>
</tbody>
</table>

Number of additions: $\approx n(n-1)(n-2)2^{n-2}$

Number of comparisons: $\approx n(n-1)(n-2)2^{n-2}$

$O(n^32^{n-2})$
Lower Bound (One)

- Lemma 7: a lower bound \( \min_{i=1,...,n} l_i + \sum_{j=1}^{n} p_j + \min_{i=1,...,n} u_i \)
- Lemma 8: a lower bound \( \sum_{j=1}^{n} (l_j + u_j) \)

- Theorem 2: \( \max \{ \min_{i=1,...,n} l_i + \sum_{j=1}^{n} p_j + \min_{i=1,...,n} u_i, \sum_{j=1}^{n} (l_j + u_j) \} \) is a lower bound
Two-phase Heuristic Algorithm

- **Constructive phase:**
  - Form a solution in a recursive manner by trying to insert one or more unscheduled job into one or more positions of a partial schedule until all jobs are sequenced
  - Two constructive heuristics are proposed:
    - Constructive Algorithm Selection (CAS)
    - Constructive Algorithm Insertion (CAI)

- **Improvement phase:**
  - Improve the quality of the solution iteratively from an initial sequence
  - Modified neighborhood search algorithm
Constructive Algorithm: CAS Heuristic

- Select the summation of a job’s loading time and a job’s unloading time closest to a given processing time

- An example:

<table>
<thead>
<tr>
<th>Job</th>
<th>$l_j$</th>
<th>$p_j$</th>
<th>$u_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>
Constructive Algorithm: CAS Heuristic (con’t)

■ 1st iteration

1st iteration

L/U Station

CNC

Job 1

Job 3

2nd iteration

2nd iteration

L/U Station

CNC

Job 1

Job 5
Constructive Algorithm: CAI Heuristic

- Each job has two weights associated with its loading and unloading times.
- A larger loading time has a larger weight. The same rule applies to unloading times.
- A job with larger sum of these two weights is selected first and it will be inserted in every position of the current sequence.
- The selected job is sequenced in the position which yields a minimum makespan.
Constructive Algorithm: CAI Heuristic (con’t)

- An example:

<table>
<thead>
<tr>
<th>Job</th>
<th>$l_j$</th>
<th>$w_{lj}$</th>
<th>$p_j$</th>
<th>$u_j$</th>
<th>$w_{uj}$</th>
<th>$w_{lj} + w_{uj}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Position 1, MS=20

5 4 3 4
3 7

Position 2, MS=18

4 5 4 3
7 3
Improvement Phase

- Neighborhood search
  - Given an initial solution, generate a set of solutions based on the initial solution
  - Adjacent pairwise interchange
    - \( n-1 \) neighborhood solutions
  - Pairwise interchange
    - \( n(n-1)/2 \) neighborhood solutions
- The neighbor with the best value is selected as the initial solution for next iteration
- The search algorithm terminates when no better neighbors can be found
Modified Neighborhood Search Algorithm

- Random selection
  - If no better neighbors can be found, randomly select a neighbor whose makespan is equal to the current best makespan as the initial solution
  - The search algorithm terminates when the number of random selections performed is greater than a threshold

- Pairwise interchange
Computational Analysis

- Two two-phase heuristic algorithms
  - CAS_M
  - CAI_M
- Problem sizes: small, medium, large
- Three settings on operations times
  - $E(p) = E(l) + E(u)$ (E: expected value)
  - $E(p) > E(l) + E(u)$
  - $E(p) < E(l) + E(u)$
Testing Plan

- Nine cases and ten runs for each case

<table>
<thead>
<tr>
<th>Number of jobs ((n))</th>
<th>Small size</th>
<th>Medium size</th>
<th>Large size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario I ((7, 11, 3))</td>
<td>((l_j, p_j, u_j) = (U(1,7), U(1, 11), U(1, 3)))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scenario II ((7, 15, 3))</td>
<td>((l_j, p_j, u_j) = (U(1,7), U(1, 15), U(1, 3)))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scenario III ((10, 11, 4))</td>
<td>((l_j, p_j, u_j) = (U(1,10), U(1, 11), U(1, 4)))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*U denotes the uniform distribution and all operation times are integer.*
### Computational Results

<table>
<thead>
<tr>
<th>Scenario</th>
<th>n</th>
<th>DP</th>
<th>CAS_M</th>
<th>CAI_M</th>
<th>LB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Optimum</td>
<td>Time</td>
<td>S1</td>
<td>S2</td>
</tr>
<tr>
<td>I (7, 11, 3)</td>
<td>10</td>
<td>68.9</td>
<td>0.06</td>
<td>74.4</td>
<td>69.3</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>121.8</td>
<td>344.8</td>
<td>136.0</td>
<td>122.6</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-</td>
<td>-</td>
<td>280.7</td>
<td>253.2</td>
</tr>
<tr>
<td>II (7, 15, 3)</td>
<td>10</td>
<td>82.2</td>
<td>0.06</td>
<td>86.8</td>
<td>82.4</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>163.1</td>
<td>356.1</td>
<td>171.4</td>
<td>163.2</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-</td>
<td>-</td>
<td>333.1</td>
<td>314.7</td>
</tr>
<tr>
<td>III (10, 11, 4)</td>
<td>10</td>
<td>80.2</td>
<td>0.07</td>
<td>85.2</td>
<td>80.9</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>155.2</td>
<td>354.4</td>
<td>162.0</td>
<td>155.6</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-</td>
<td>-</td>
<td>335.9</td>
<td>323.0</td>
</tr>
</tbody>
</table>

(S1: the constructive stage; S2: the improvement stage)
Computational Results (con’t)

- Relative error (RE)

<table>
<thead>
<tr>
<th>Scenario</th>
<th>$n$</th>
<th>RE from optimum (%)</th>
<th>RE from LB (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CAS M</td>
<td>CAI M</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SI</td>
<td>S2</td>
</tr>
<tr>
<td>I (7, 11, 3)</td>
<td>10</td>
<td>8.12</td>
<td>0.64</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>11.73</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>II (7, 15, 3)</td>
<td>10</td>
<td>5.73</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>5.37</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>III (10, 11, 4)</td>
<td>10</td>
<td>6.17</td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>4.51</td>
<td>0.28</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Computational Results (con’t)

- Both algorithms (CAS_M, CAI_M) can obtain a solution in one second.
- The quality of solutions obtained by CAS_M and CAI_M are similar.
- Scenario I, on average, the solution from the optimum is 1.1% and from the lower bound is 3%.
- CAI outperforms CAS.
Agenda

- Introduction
- T-line machining center with one machine
  - Complexity
  - Common unloading times
  - Dynamic programming algorithm
  - Lower bound
  - Heuristic algorithms and computational results
- T-line machining center with $m$ machines
  - Dynamic programming algorithm
  - Lower bound
- Conclusions and future research
T-line Machining Center with $m$ Machines

- The rotary table with $m+1$ pallets
DP Algorithm \((m)\)

Optimal value function (OVF): \(f_i(S, \Psi, j) = \text{minimum completion time for processing jobs in} \Psi \text{ on machines, unloading the last job in } S \text{ and loading job } j \text{ at the L/U station, given that the } i \text{ jobs in } S \text{ have already been completed.} \) (5.10)
DP Algorithm ($m$) (cont' t)

Recurrence relation (RR):

$$f_i(S, \Psi, j) = \min_{k \in S} \{f_{i-1}(S \setminus \{k\}, \{k\} \cup \Psi \setminus \Psi_{(m)}, \Psi_{(m)}) + \max \{p_{\Psi_{(m)}}, p_{\Psi_{(m-1)}}, \ldots, p_{\Psi_{(1)}}, u_k + l_j\}\};$$

$$i = 1, \ldots, n - m - 1; \{\Psi, j\} \subseteq N; S \subseteq N \setminus \{\Psi, j\}, |S| = i. \quad (5.12)$$

Boundary condition (BC):

$$f_0(\emptyset, \Psi, j) = \sum_{i=1}^{m} \max \{p_{\Psi_{(i-1)}}, p_{\Psi_{(i-2)}}, \ldots, p_{\Psi_{(i-m)}}, l_{\Psi_{(i)}}\} + \max \{p_{\Psi_{(m)}}, p_{\Psi_{(m-1)}}, \ldots, p_{\Psi_{(1)}}, l_j\}; \{\Psi, j\} \subseteq N. \quad (5.13)$$
DP Algorithm \((m)\) (con’t)

Answer (ANS): \(\min_{\Psi \subseteq N} \{ f_{n-m}(S, \Psi, \emptyset) \} \)  \hspace{1cm} (5.14)

where \( f_{n-m}(S, \Psi, \emptyset) = \min_{k \in S} \{ f_{n-m-1}(S \setminus \{k\}, \{k\} \cup \Psi \setminus \Psi_{(m)}, \Psi_{(m)}) + \max \{ p_{\Psi_{(m)}1}, \ldots, p_{\Psi_{(m)}m}, u_k \} \} \)

\[ + \sum_{i=1}^{m} \max \{ p_{\Psi_{(i-m)}1}, p_{\Psi_{(i-m)}2}, \ldots, p_{\Psi_{(i)}m}, u_{\Psi_{(i)}} \} ; \Psi \subseteq N ; S = N \setminus \Psi , |S| = n - m. \]  \hspace{1cm} (5.15)
**DP Algorithm (m) - Computational Analysis**

<table>
<thead>
<tr>
<th>Stage</th>
<th>Number of combinations</th>
<th>Addition</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC ( (i = 0) )</td>
<td>((m+1)!C_{m+1}^n)</td>
<td>(m)</td>
<td>(m(m+1)/2)</td>
</tr>
<tr>
<td>RR ( (1 \leq i \leq n-m-1) )</td>
<td>((m+1)!C_{m+1}^n C_{i}^{n-(m+1)})</td>
<td>(2i)</td>
<td>(m \times i + (i - 1))</td>
</tr>
<tr>
<td>ANS ( f_{n-m}(i = n-m) )</td>
<td>(m! C_{m}^n)</td>
<td>(n)</td>
<td>(m(n-m) + (n-m-1) + m(m-1)/2)</td>
</tr>
<tr>
<td>Min makespan</td>
<td>1</td>
<td>0</td>
<td>(m! C_{m}^n - 1)</td>
</tr>
</tbody>
</table>

**Number of additions:**

\[
\approx \frac{n!}{(n-m-2)!} 2^{n-m-1}
\]

**Number of comparisons:**

\[
\approx \frac{n!(m+1)}{(n-m-2)!} 2^{n-m-2}
\]

\[
O\left(\frac{n!m}{(n-m-2)!} 2^{n-m-2}\right)
\]
Lower Bound \((m)\)

- Consider a relaxed problem
  - Neglecting loading and unloading times
  - Relax the job order processed by machines
  - Denote the optimal makespan of the relaxed problem as \(LB_R\)
Lower Bound \((m)\) (con’t)

Algorithm for \(LB_R\)

- Sort processing times in descending order

<table>
<thead>
<tr>
<th>Job</th>
<th>(p_{j1})</th>
<th>(p_{j2})</th>
<th>(p_{j3})</th>
<th>(p_{j4})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>8</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>15</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>4</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>13</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>14</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>11</td>
<td>8</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>14</td>
<td>15</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Set</th>
<th>(p_{j1})</th>
<th>(p_{j2})</th>
<th>(p_{j3})</th>
<th>(p_{j4})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R_1)</td>
<td>14</td>
<td>15</td>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>(R_2)</td>
<td>11</td>
<td>15</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>(R_3)</td>
<td>11</td>
<td>14</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>(R_4)</td>
<td>9</td>
<td>13</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>(R_5)</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>(R_6)</td>
<td>7</td>
<td>8</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>(R_7)</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>(R_8)</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Lower Bound \((m)\) (con’t)

Algorithm for \(LB_R\)

- Place the largest processing time on each machine to cycle 5, the second largest to cycle 6, and so on

![Diagram showing processing times on various machines across cycles 1, 5, 9, and 13.]

- L/U Station
- CNC1
- CNC2
- CNC3
- CNC4

\(R_1\), \(R_2\), \(R_3\), \(R_4\), \(R_5\)

\(u_{[13]}\)
Lower Bound \((m)\) (con’t)

Algorithm for \(\text{LB}_R\)

- Assign the processing time on machine 1 in \(R_6\) to cycle 4
- Assign the processing time on machine 4 in \(R_6\) to cycle 10
Lower Bound ($m$) (con’t)

Algorithm for LB$R$

- Assign the processing time on machine 1 in $R_7$ to cycle 3
- Assign the processing times on machines 3 and 4 in $R_7$ to cycles 10 and 11, respectively

<table>
<thead>
<tr>
<th>L/U Station</th>
<th>$l_{[1]}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNC1</td>
<td>5 7 14 11 11 9 8</td>
</tr>
<tr>
<td>CNC2</td>
<td>8 15 15 14 13 8 4</td>
</tr>
<tr>
<td>CNC3</td>
<td>6 13 9 8 8 7 6</td>
</tr>
<tr>
<td>CNC4</td>
<td>15 11 11 10 7 3 2</td>
</tr>
</tbody>
</table>

Cycle 1    Cycle 5    Cycle 9    Cycle 13
Lower Bound ($m$) (con’t)

Algorithm for $LB_R$

- Assign the processing times in $R_8$ to each available cycle and the makespan is 91
Lower Bound ($m$) (con’t)

- **Lemma 9**: $LB_R$ is the optimal makespan for the relaxed problem

- Ideas of the proof: interchange any two processing times on a machine will not reduce $LB_R$

- **Theorem 3**: \[
\max\left\{ \sum_{j=1}^{n} (l_j + u_j), \min_{i=1,\ldots,n} l_i + LB_R + \min_{i=1,\ldots,n} u_i \right\}
\]
is a lower bound
Agenda

- Introduction
- T-line machining center with one machine
  - Complexity
  - Common unloading times
  - Dynamic programming algorithm
  - Lower bound
  - Heuristic algorithms and computational results
- T-line machining center with \( m \) machines
  - Dynamic programming algorithm
  - Lower bound
- Conclusions and future research
Conclusions

- Synchronous transportation times:
  - The problem has been showed to be NP-hard
  - A DP algorithm has been proposed for a general case with \( m \) machines
  - Two-phase heuristic algorithms have been developed to obtain near optimal solutions in a very short time for the problems with one machine
  - Lower bounds are derived for the cases with one and \( m \) machines
  - An algorithm integrated the G-G algorithm has been provided to solve the one-machine problem with a common loading or unloading time
Future Research

- Mateheuristics
- Common processing times for the one-machine case
- For two-machine case
  - Unloading times are zero
  - Processing times on one machine dominate those on another machine
- Different objectives
Thank you.
Questions?
Lemma 1

The total processing time on the CNC machine is $Z - 2$

$$\sum_{i=1}^{n} (p(Ja_i) + p(Jb_i) + p(Jc_i)) = 3n + n(\beta + 1) + \sum_{i=1}^{n} (3\beta + 3c_i)$$

$$= 4n + 4n\beta + \sum_{i=1}^{n} 3c_i = 4n + 4n\beta + 3C_0 = Z - 2$$

Since the smallest loading and unloading times are equal to 1 respectively, the idle time can only occur at the first and last cycles on the CNC machine.
Lemma 2:

The total operation time on the L/U station is \( Z - 2 \)

\[
\sum_{i=1}^{n} [l(Ja_i) + l(Jb_i) + l(Jc_i) + u(Ja_i) + u(Jb_i) + u(Jc_i)]
\]

\[
= 4n + 4n\beta + 3\sum_{i=1}^{n} (a_i + b_i) = 4n + 4n\beta + 3C_0 = Z - 2
\]

One unit of idle time can only occur in the second and second last cycles at the L/U station.
Lemma 5: $J_0$ is sequenced in the first position

\[ Z^1 - Z^2 = l_a + p_b + p_c - \max(l_a, p_c) - p_b = l_a + p_c - \max(l_a, p_c) > 0 \]
Lemma 6: $J_{n+1}$ is sequenced in the last position

\[
Z^1 - Z^2 = l_a + \max(c, p_b) + l_c + p_d - l_c - \max(l_a, p_d) - \max(c, p_b)
\]

\[
= l_a + p_d - \max(l_a, p_d) > 0.
\]
Lemma 9

- Let these two processing times on machine $k$ in cycles $r$ and $s$ be $p_k^r$ and $p_k^s$, respectively.
- Assume that $C_r \leq C_s$, then $p_k^r \leq p_k^s$.
- Interchange these two processing times.
Lemma 9 (cont’t)

- Assume that \( C_r = p_k^r, \ C_s = p_k^s \)
- After interchange,

\[
C_r' = p_k^s = C_s \quad \text{and} \quad C_r = p_k^r \leq C_s'
\]

\[
C_r + C_s \leq C_r' + C_s'
\]
Lemma 9 (con’t)

- Assume that \( C_r = p_r^r, \ C_s > p_s^s \)
- After interchange,

\[
C_r \leq C'_r = p^s_k \quad \text{and} \quad C_s = C'_s
\]

\[
C_r + C_s \leq C'_r + C'_s
\]
Lemma 9 (con’t)

- Assume that $C_r > p^r_k$, $C_s = p^s_k$

- After interchange,

$$C'_r = C'_s = p^s_k$$

$$C'_s = \max\{p^r_k, \max_{f \neq k}\{p^s_f\}\}$$

(without considering $p^s_k$ and $p^r_k$, $\max_{f \neq k}\{p^s_f\} \geq \max_{h \neq k}\{p^r_h\} > p^r_k$)

$$C'_s = \max_{f \neq k}\{p^s_f\} \geq C_r$$

$$C_r + C_s \leq C'_r + C'_s$$
Lemma 9 (con’t)

- Assume that $C_r > p_k^r$, $C_s > p_k^s$
- After interchange,

$$C'_r = \max\{C_r, p_k^s\} \geq C_r \quad \text{and} \quad C_s = C'_s$$

$$C_r + C_s \leq C'_r + C'_s$$