A variable neighbourhood search for hybrid flow-shop scheduling problem with rework and set-up times.


July. 10, 2018

Presented by Yooney Cho
Introduction

Keywords

- Motivated from final inspection system in automotive manufacturing system
- Hybrid flow-shop scheduling problem
- Inspection in the last stage for testing defective jobs
- Rework (re-entrant flow) for defective jobs
- Minimizing makespan
- MIP model
- Two heuristic solution algorithms (Rule based, Variable neighbourhood search)
## Literature review

### System Considerations

<table>
<thead>
<tr>
<th>System</th>
<th>Considerations</th>
<th>Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Re-entrant flow for only defected jobs (rework)</td>
<td>Rabiee (2012)-considered rework in no-wait two machine flow shop, at each stage, an inspection is considered after job processing</td>
</tr>
<tr>
<td></td>
<td>Re-entrant flow for only defected jobs (rework)</td>
<td>-</td>
</tr>
</tbody>
</table>

### Contribution

- First research on hybrid flow shop scheduling with reworks
- First to employ VNS method for hybrid flow shop scheduling
Problem description

- Configuration
  - Two cases of the problem
    - A: Defective jobs must be processed in all stages again
    - B: Defective jobs are only processed in some stages

*Job cannot rework more than once!*

**Figure 1** Flow of jobs in (a) shop number 1 and (b) shop number 2.
Problem description

Assumptions

- Unlimited buffers between the stages
- Negligible transportation times
- Unrelated parallel machines
- No machine breakdowns

- Each job $i$ arrives at ready time $r_i$
- Setup times are sequence dependent
- No pre-emption
- All of the problem data are deterministic and given

- Failure of job $i$ has distribution with parameter $\theta_i$
- Job cannot rework more than once
Problem description

- Problem description
  - Decision variables
    - Machine assignment
    - Job sequence

- Objective
  - Minimizing makespan

- Constraints
  - -
Solution approach

- **Mathematical model**
  - Extended form of the model by Jungwattanankit et al (2008)
  - **Notation**

\[
\begin{align*}
  i, i' & \text{ indices for jobs, } i, i' = 1, 2, \ldots, n \ (i \neq i') \\
  j, j' & \text{ indices for operations, } j, j' = 1, 2, \ldots, e_i \ (j \neq j'), \ e_i \text{ denotes the expected number of operations for job } i, \ e_i = m/(1-\theta_i) \\
  t & \text{ index for stage, } t = 1, 2, \ldots, m \\
  k_t & \text{ index for machine at stage } t, \ k_t = 1, 2, \ldots, M_t \\
  O_{ij} & \text{ operation } j \text{ of job } i \\
  St(j) & \text{ stage where the operation } O_{ij} \text{ is performed} \\
  p_{ijk} & \text{ processing time of operation } O_{ij} \text{ on machine } k \\
  S_{it}^t & \text{ set-up time between job } i \text{ and job } i' \text{ at stage } t \\
  r_i & \text{ ready time for job } i \\
  B & \text{ large positive number} \\
  C_{ij} & \text{ completion time of operation } O_{ij} \\
  Y_{ij}^{k,j} & \text{ 1 if operation } O_{ij} \text{ is scheduled immediately after operation } O_{ij'} \text{ on machine } k \text{ at stage } St(j), \text{ and 0 otherwise}
\end{align*}
\]
Solution approach

- **Mathematical model**
  - **Constraints**
    - Every operation of each job should have one predecessor
    - Every operation of each job should have one successor
    
    \[
    \sum_{k \in S(i)} \sum_{i=0}^{n} y_{ij}^k = 1, \quad \forall i, j
    \]
    \[
    \sum_{k \in S(j)} \sum_{i'=1}^{n+1} \sum_{j'=1}^{e_j} y_{ij'}^k = 1, \quad \forall i, j
    \]
Solution approach

- **Mathematical model**
  - **Constraints**
    - Each operation should be performed at each stage only once.
    
    \[ Y_{i,j}^k = 0, \quad \forall \, i, j, k \in St(j) \]

    Indices for operations, \( j, j' = 1, 2, \ldots, e_i \) \((j \neq j')\), \(e_i\) denotes the expected number of operations for job \(i\), \(e_i = m/(1-\theta_i)\)
Solution approach

- Mathematical model
  - Constraints

  - Assign only one operation to the first and last positions on each machine at each stage.

  \[
  \sum_{i=1}^{n+1} \sum_{j=1}^{e_i} y_{0ijj'}^k = 1, \quad \forall j, k \in St(j) \\
  \sum_{i=0}^{n} \sum_{j=1}^{e_i} y_{ij,n+1,j'}^k = 1, \quad \forall j', k \in St(j')
  \]

  Job 0 = dummy job

  Operation $O_{0j}$

  Operation $O_{ij}$

  Job $n+1$ = dummy job

  Operation $O_{n+1,j'}$
Solution approach

- **Mathematical model**
  - **Constraints**
    - Forces to contrast a sequence at each stage.

\[
\sum_{i'=1}^{n+1} \sum_{j'=1}^{\xi_i} Y_{i'j'}^{k} = \sum_{i'=0}^{n} \sum_{j'=1}^{\xi_i} Y_{i'j'}^{k},
\forall i,j,k \in S(t(j))
\]
Solution approach

- **Mathematical model**
  - **Constraints**
    - If operation $O_{ij}$ is scheduled immediately after operation $O_{i'j'}$ on the same machine at a particular stage, then processing of operation $O_{i'j'}$ must be finished before beginning of operation $O_{ij}$.
    
    $C_{ij} \geq C_{i'j'} + S_{ij}^{S_{ij'}} + p_{ijk} + (Y_{i'j'}^k - 1)B,$
    
    $\forall i, j, i'j', k \in S_{ij'}(j)$

    - Precedence constraints for the jobs and ensures that a job cannot start its $j$ operation before finishing operation $j-1$.
    
    $C_{ij} \geq C_{ij-1} + \sum_{k \in S_{ij}(j)} \sum_{i'j'=i}^{n} \sum_{j'=0}^{i-1} Y_{i'j'j'}^k (S_{ij'}^{S_{ij'}} + p_{ijk}),$
    
    $\forall i, j \geq 2$

If $Y_{i'j'j'}$ is $1$ then, $C_{ij} \geq C_{i'j'} + S_{ij'} + P_{ijk}$. 

Diagram showing the scheduling of operations $O_{i'j'}$ and $O_{ij}$ on a machine $k$.
Solution approach

- **Mathematical model**
  - **Constraints**
    - Completion time of operation 1 at the first stage is greater than or equal to ready time plus processing and set-up time at first stage.
      \[
      C_{i1} \geq r_i + \sum_{k \in S(i)} \sum_{i' = 0}^{n} \sum_{j' = 1}^{p} Y_{i'j'i1}^k (S_{i'i}^{(1)} + P_{i1k}), \forall i
      \]
    - Conditions of decision variable and completion time.
      \[
      C_{\text{max}} \geq C_{i1}, \quad \forall i \\
      C_{ij} \geq 0, \quad \forall i, j \\
      Y_{ij'j'i}^k \in \{0, 1\}, \quad \forall i, j, i', j', k
      \]

If \( Y_{i'j'i1} \) is 1 then, \( C_{i1} \geq r_i + S_{i'i} + P_{i1k} \)
Solution approach

- Mathematical model
  - Model

\[
\begin{align*}
\text{Minimize} & \quad C_{\text{max}} \\
\text{Subject to} & \\
\sum_{k \in S(t)} \sum_{i=0}^{s_i} \sum_{j=1}^{s_j} y_{ij}^k &= 1, \quad \forall i, j^{'} \\
\sum_{k \in S(t)} \sum_{j=1}^{s_j} \sum_{i=0}^{s_i-1} y_{ij}^k &= 1, \quad \forall i, j \\
y_{ii}^k &= 0, \quad \forall i, j, k \in S(t(j)) \\
\sum_{t=1}^{n+1} \sum_{j=1}^{s_j} y_{0j}^k &= 1, \quad \forall j, k \in S(t(j)) \\
\sum_{i=0}^{n} \sum_{j=1}^{s_j} y_{ij,n+1,j}^k &= 1, \quad \forall j, k \in S(t(j)) \\
\sum_{t=1}^{n+1} \sum_{j=1}^{s_j} y_{ij,0,j}^k &= \sum_{i=0}^{n} \sum_{j=1}^{s_j} y_{ij,j}^k, \quad \forall i, j, k \in S(t(j)) \\
C_{ij} &\geq C_{ij} + S_{ij}^{(0)} + p_{ijk} + (Y_{ij}^k - 1)B, \quad \forall i, j, j, k \in S(t(j)) \\
C_{ij} &\geq C_{ij} + \sum_{k \in S(t(j))} \sum_{t=0}^{n} \sum_{j=1}^{s_j} y_{ij,j}^k (S_{ij}^{(0)} + p_{ijk}), \quad \forall i, j \geq 2 \\
C_{ij} &\geq n + \sum_{k \in S(t(j))} \sum_{t=0}^{n} \sum_{j=1}^{s_j} y_{ij,j}^k S_{ij}^{(1)} + p_{ijk}), \quad \forall i \\
C_{\text{max}} &\geq C_{\text{max}i} \quad \forall i \\
C_{ij} &\geq 0, \quad \forall i, j \\
y_{ij}^k &\in \{0, 1\}, \quad \forall i, j, j^{'} , j^{''}, k
\end{align*}
\]
Solution approach

- Solution algorithms (1)
  - Heuristic algorithm

- Assign the sorted jobs to the machine that completes the job earlier.

- For stage 2 to last stage, jobs according to their completion time at the previous stage are assigned to the machine that completes the job earlier.

- First, sort the jobs based on selected dispatching rules for the first stage.

- 5 alternative dispatching rules; SPT, LPT, ERT, SRPT, LRPT
Solution approach

- Solution algorithms (1)
  - Heuristic algorithm

SPT (shortest processing time): sorts the jobs in non-decreasing order of the processing times. In the single-machine scheduling problem, using the SPT rule will be minimizing the sum of completion times.
LPT (longest processing time): sorts the jobs in non-increasing order of the processing time. LPT rule tends to balance the workload over the machines in parallel machines scheduling problem.
ERT (earliest ready time): sorts the jobs according to arrival time of jobs to shop. ERT rule is equivalent to the first-in first-out (FIFO) rule.
SRPT (shortest remaining processing time): sorts the jobs in non-decreasing order of sum of remaining processing times (sum of processing times of the successor operation including itself). SRPT rule is an extended form of SPT rule for classical flow-shop.
LRPT (longest remaining processing time): sorts the jobs in non-increasing order of sum of remaining processing times. LRPT rule is an extended form of LPT rule for classical flow-shop.
Solution approach

- Solution algorithms (2)
  - Variable neighbourhood search (VNS)

- First, assign the jobs to the machine that has the shortest queue or shortest processing time.

- Second, sort the jobs based on selected dispatching rules for that machine.

- 8 alternative dispatching rules per each machine; SPT, LPT, SRPT, LRPT, FIFO, LIFO, PTRPT, TIS
Solution approach

- Solution algorithms (2)
  - Variable neighbourhood search (VNS)

<table>
<thead>
<tr>
<th>Array</th>
<th>Rule</th>
<th>Rule definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>First-In First-Out</td>
<td>The first arrived job is served first</td>
</tr>
<tr>
<td>2</td>
<td>Last-In First-Out</td>
<td>The first arrived job is served first</td>
</tr>
<tr>
<td>3</td>
<td>Shortest Processing Time</td>
<td>The job with the shortest processing time is served first</td>
</tr>
<tr>
<td>4</td>
<td>Longest Processing Time</td>
<td>The job with the longest processing time is served first</td>
</tr>
<tr>
<td>5</td>
<td>Shortest Remaining Processing Time</td>
<td>The job with the shortest processing times of remaining operations is served first</td>
</tr>
<tr>
<td>6</td>
<td>Longest Remaining Processing Time</td>
<td>The job with the longest processing times of remaining operations is served first</td>
</tr>
<tr>
<td>7</td>
<td>Processing Time to Remaining</td>
<td>The job with the smallest ratio of processing time to processing times of remaining operations is served first</td>
</tr>
<tr>
<td>8</td>
<td>Time In System</td>
<td>The job with the longest time present in the shop is served first</td>
</tr>
</tbody>
</table>
Computational experiment

- **Design of experiments (1)**
  - **Equipment**
    - Software package: CPLEX 12.1
    - PC with Intel processor operating at 2.53 GHz
  - **Test data**

<table>
<thead>
<tr>
<th>Number of stages</th>
<th>Number of jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3,4,5,6,10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of machines per stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1, 2-2</td>
</tr>
<tr>
<td>1-1-2</td>
</tr>
</tbody>
</table>

- Number of rework jobs(failure) ~ 2 → deterministic rework to evaluate feasibility of model
- Processing time ~ DU(20, 120)
- Setup times ~ DU(5, 25)
- Ready times ~ DU(0, 30)
- ? instance for each of 15 combinations
Computational experiment

- Test result (1)
  - Discussion
    - Using the mathematical model to solve the problem is not efficient.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Number of stages</th>
<th>Number of machines per stage</th>
<th>Number of jobs</th>
<th>CPLEX results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2–1</td>
<td>3</td>
<td>optimal</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2–1</td>
<td>4</td>
<td>optimal</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2–1</td>
<td>5</td>
<td>optimal</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2–1</td>
<td>6</td>
<td>optimal</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2–1</td>
<td>10</td>
<td>Gap: infinite</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2–2</td>
<td>3</td>
<td>optimal</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2–2</td>
<td>4</td>
<td>optimal</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>2–2</td>
<td>5</td>
<td>optimal</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>2–2</td>
<td>6</td>
<td>Gap: 9.85%</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>2–2</td>
<td>10</td>
<td>Gap: 48.80%</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>1–1–2</td>
<td>3</td>
<td>optimal</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>1–1–2</td>
<td>4</td>
<td>optimal</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>1–1–2</td>
<td>5</td>
<td>optimal</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>1–1–2</td>
<td>6</td>
<td>Gap: 37.48%</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>1–1–2</td>
<td>10</td>
<td>Gap: infinite</td>
</tr>
</tbody>
</table>
Computational experiment

- Design of experiments (2)
  - Equipment
    - Algorithms are coded in MATLAB
    - PC with Intel processor operating at 2.53 GHz
  - Test data

<table>
<thead>
<tr>
<th>Number of stages</th>
<th>Number of jobs</th>
<th>Failure probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>20,40,60,80,100</td>
<td>0.1, 0.2, 0.3</td>
</tr>
</tbody>
</table>

- Processing time ~ DU(20, 120)
- Setup times ~ DU(5, 25)
- Ready times ~ DU(0, 30)
- Number of machines per each stage ~ DU(1, 4)
Test result (2)

Discussion

- VNS algorithm worked better than heuristic algorithm in both rework problems
- Among heuristic rules, SPT rule worked better than other rules

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Problem number 1</th>
<th>Problem number 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average RDI</td>
<td>$n_p$</td>
</tr>
<tr>
<td>Heuristic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERT</td>
<td>0.514</td>
<td>1</td>
</tr>
<tr>
<td>SPT</td>
<td>0.321</td>
<td>2</td>
</tr>
<tr>
<td>LPT</td>
<td>0.932</td>
<td>0</td>
</tr>
<tr>
<td>SRPT</td>
<td>0.540</td>
<td>1</td>
</tr>
<tr>
<td>LRPT</td>
<td>0.465</td>
<td>0</td>
</tr>
<tr>
<td>Metaheuristic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VNSq</td>
<td>0.063</td>
<td>7</td>
</tr>
<tr>
<td>VNSp</td>
<td>0.264</td>
<td>4</td>
</tr>
</tbody>
</table>

test problem, the relative deviation index for algorithm $h$ is defined as $RDI = (C_h - C_{best}) / (C_{worst} - C_{best})$, where $C_h$ is the objective function value obtained by algorithm $h$, and $C_{best}$ and $C_{worst}$ are the best and the worst objective function value, respectively, among those obtained from all of the algorithms.
Computational experiment

- Test result (2)
  - Discussion
    - VNSp slightly worked better than VNSq in the 3 stage problems
    - VNSq becomes much better than VNSp as the number of stages grows
      - VNSp - unbalanced workload over machines
      - VNSq - balanced workload over machines

Figure 4 Comparison of VNSq and VNSp for (a) shop number 1 and (b) shop number 2.
Conclusions

- **Disadvantage**
  - Design of experiment (CPLEX test)
    - No data for instance number per each combinations
    - Number of rework jobs ~ 2
    - No runtime data

- **Advantage**
  - Contribution
    - First research on hybrid flow shop scheduling with reworks

- **Design of experiments (1)**
  - Equipment
    - Software package: CPLEX 12.1
    - PC with Intel processor operating at 2.53 GHz
  - Test data
    
    | Number of stages | Number of Jobs |
    |------------------|---------------|
    | 2                | 3             |
    | 3, 4, 5, 6, 10   |               |

    | Number of machines per stage |
    |-------------------------------|
    | 1-1-2                         |

- **Literature review**

<table>
<thead>
<tr>
<th>System</th>
<th>Considerations</th>
<th>Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Re-entrant flow for only detected jobs (rework)</td>
<td>-</td>
</tr>
</tbody>
</table>

- **Contribution**
  - First research on hybrid flow shop scheduling with reworks
  - First to employ VNS method for hybrid flow shop scheduling
Thank You!