A hybrid algorithm for flexible job-shop scheduling problem with setup times

Amei Azzouz, Meriem Ennigrou and Lamjed Ben Said
International Journal of Production Management and Engineering, 2017
1. Introduction

**Flexible Jobshop scheduling problem (FJSP)**

- FJSP which is an extension of the classical JSP allows one operation to be processed by any machine from a given set

Set $J = \{J_1, J_2, \ldots, J_n\}$ of independent jobs

$I_j$ is formed by a sequence $O_{i1}, O_{i2}, \ldots, O_{ni}$ of operations

Set $U = \{M_1, M_2, \ldots, M_m\}$ of machines

Each operation $O_{ij}$ can be executed on any among a subset $U_{ij} \subseteq U$ of compatible machines

- FJSP can be decomposed into two sub-problems
  I. The machine selection problem (resource assignment problem)
  II. Operations sequencing problem

- FJSP is a well-known NP-hard problem (Garey et al, 1976)

Many researches have been made to find the near optimal solution of FJSP

**Set-up times**

- Many real-life situations such as chemical, printing, pharmaceutical and automobile manufacturing set-up times are required between jobs

- Set-up times are also strongly dependent on job itself (sequence independent) and the previous job that ran on the same machine (sequence dependent)

- Reducing the setup times is an important task to improve shop performance
## Literature review

<table>
<thead>
<tr>
<th>Problem</th>
<th>Author</th>
<th>Year</th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDST-FJSP</td>
<td>Imanipour</td>
<td>2006</td>
<td>Non linear mixed integer programming model and Tabu search</td>
</tr>
<tr>
<td></td>
<td>Saidi-Mehrabad and Fattaihi</td>
<td>2007</td>
<td>Tabu-Search considering operations which can be performed by two machines</td>
</tr>
<tr>
<td></td>
<td>Bagheri and Zandieh</td>
<td>2011</td>
<td>Integrated approach, artificial immune system algorithm, particle swarm optimization to minimize Aggregate objective function (AOF)</td>
</tr>
<tr>
<td></td>
<td>Sadrzadeh</td>
<td>2013</td>
<td>AOF = αF1 + (1−α)F2</td>
</tr>
<tr>
<td></td>
<td>Oddi et al</td>
<td>2011</td>
<td>Iterative flattering search and propose a new benchmark problems</td>
</tr>
<tr>
<td></td>
<td>Mousakhani</td>
<td>2013</td>
<td>Mixed integer linear programming model and meta-heuristic based on iterated local search</td>
</tr>
<tr>
<td></td>
<td>Gonzalez et al</td>
<td>2013</td>
<td>Memetic algorithm which consists of Tabu search and Genetic algorithm</td>
</tr>
<tr>
<td></td>
<td>Rossi</td>
<td>2014</td>
<td>SDST-FJSP with transportation times using ant-colony algorithm</td>
</tr>
<tr>
<td></td>
<td>Allahverdi</td>
<td>2015</td>
<td>Reviews about 500 papers on scheduling problems with setups</td>
</tr>
</tbody>
</table>

- In this paper, New contribution to SDST-FJSP
  - Hybrid genetic algorithm (HGA) based on GA and VNS shows that this algorithm can be very effective with respect to the state of the art
2. Problem description

- **Objective function:**
  - Minimizing the makespan
  - Minimizing the Aggregate objective function (AOF)
    - AOF = αF₁ + (1-α)F₂, where α denote the weight
    - F₁ = makespan, F₂ = mean tardiness

- **Decision Variable:**
  - The assignment of operations to the machines
  - The sequencing of operations on the machines

- **Assumption:**
  - Jobs are independent of each other
  - Machines are independent of each other
  - One machine can process at most one operation at a time
  - No preemption is allowed
  - All jobs and machines are available at time 0
  - Setup times are dependent on the sequence of jobs
3. Solution approach

- **The hybrid genetic algorithm (HGA)**
  - GA is well known as powerful methods for solving combinatorial optimization problems such as scheduling problem
  - It is combines the global search and local search

**Procedure**

- **Beginning**
- **Parameters Setting**
- **Initialize the Population**
- Evaluation = objective function
- **Termination criteria satisfied?**
  - Y: **Output the Best solution**
  - N: **Selection** Choose (N/2) using tournament selection
    - **Crossover**
    - **Mutation**
    - **VSN local search**
    - **N best solution choose**
- **Ending**
3. Solution approach

1) Initialize the population

- Solution representation – Binary matrix
  - rows: the sequence of operations to be processed
  - columns: the used machines

Ex)

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>O11</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>O21</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>O22</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>O12</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>O13</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>O23</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>O31</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>O32</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>O33</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Initial population of N is based on three traditional dispatching rules

I. Shortest processing time (SPT) ; 20%
II. Longest processing time (LPT) ; 20%
III. Heuristic rules based on local search algorithm ; 20%
  remaining with random solution ; 40%
3. Solution approach

Ex) Existing initialize population by localization

| Approach by localization (machine workload updates in bold) |
|-----------------|-----------------|-----------------|-----------------|-----------------|
|                 | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_1$ | $M_2$ | $M_3$ | $M_4$ |
| $O_{11}$        | 7     | 6     | 4     | 5     | 7     | 6     | 4     | 5     | 7     | 6     | 4     | 5     |
| $O_{12}$        | 4     | 8     | 5     | 6     | 4     | 8     | 9     | 6     | 4     | 8     | 9     | 6     |
| $O_{13}$        | 9     | 5     | 4     | 7     | 9     | 5     | 8     | 7     | 13    | 5     | 8     | 7     |
| $O_{21}$        | 2     | 5     | 1     | 3     | 2     | 5     | 5     | 3     | 6     | 5     | 5     | 3     |
| $O_{22}$        | 4     | 6     | 8     | 4     | 4     | 6     | 12    | 4     | 8     | 6     | 12    | 4     |
| $O_{23}$        | 9     | 7     | 2     | 2     | 9     | 7     | 6     | 2     | 13    | 7     | 6     | 2     |
| $O_{31}$        | 8     | 6     | 3     | 5     | 8     | 6     | 7     | 5     | 12    | 6     | 7     | 5     |
| $O_{32}$        | 3     | 5     | 8     | 3     | 3     | 5     | 12    | 3     | 7     | 5     | 12    | 3     |

- The initial population is generated following the approach by localization of Kacem
- This approach takes into account both the processing times and the workload of the machines
- For each operation, the machine with the minimum processing time, fixing that assignment, and then to add this time to every subsequent entry in the same column

2) Evaluation

- fitness evaluation function for the chromosomes coincides with the objective function
3. Solution approach

3) Selection

- Choose \((N/2)\) members from the population using tournament selection

4) Crossover

a. Select randomly two positions \(XP1\) and \(XP2\) in parent 1

b. The middle part is copied to the offspring 1

c. The rest of this child is filled from the parent2 starting with position \(XP2+1\), and jumping elements that are already presented in offspring1
3. Solution approach

5) Mutation

- Select an operation on the machine with the maximum workload, and assign it to the machine with the minimum workload if compatible

\[
\begin{array}{ccc}
M1 & M2 & M3 \\
O11 & 0 & 0 & 1 \\
O21 & 1 & 0 & 0 \\
O22 & 0 & 1 & 0 \\
O12 & 0 & 0 & 1 \\
O13 & 1 & 0 & 0 \\
O23 & 0 & 0 & 1 \\
O31 & 0 & 1 & 0 \\
\end{array}
\]

6) VNS algorithm

- The hybridization between GA and local search algorithm is based on simple local search which is inspired from Bagheri and Zandieh (2011) and Ennigrou and Ghedira (2008)

1. Initial solution: the same solution representation as in GA
2. Generate neighborhood
   a. Change the sequence of the operations
   b. Change the assignment of the operations to the machines
   c. The combination between the two precedent neighborhood structures
3. Neighborhood evaluation
4. The best solution founded is selected to the next population
3. Solution approach

- Example)

a. Two positions P1 and P2 are randomly selected on the solution and then, the operations between them are randomly reordered.

```
<table>
<thead>
<tr>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
</table>
O11 0  0  1
O21 1  0  0
O22 0  1  0
O12 0  0  1
O13 1  0  0
O23 0  0  1
O31 0  1  0
... ...
```

```
<table>
<thead>
<tr>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
</table>
O11 0  0  1
O21 1  0  0
O13 1  0  0
O22 0  1  0
O12 0  0  1
O13 1  0  0
O23 0  0  1
O31 0  1  0
... ...
```

b. Operation is chosen randomly and then, changing the assignment of the selected operation to another machine.

```
<table>
<thead>
<tr>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
</table>
O11 0  0  1
O21 1  0  0
O22 0  1  0
O12 0  0  1
O13 1  0  0
O23 0  0  1
O31 0  1  0
... ...
```

```
<table>
<thead>
<tr>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
</table>
O11 0  0  1
O21 1  0  0
O22 1  0  0
O12 0  0  1
O13 1  0  0
O23 0  0  1
O31 0  1  0
... ...
```
3. Solution approach

- **Example**)

  The combination between the two precedent neighborhood structures

  ┌─────────────┐
  │ M1 M2 M3    │
  ├─────────────┤
  │ O11 0 0 1   │
  │ O21 1 0 0   │
  ├─────────────┤
  │ O22 0 1 0   │
  │ O12 0 0 1   │
  │ O13 1 0 0   │
  ├─────────────┤
  │ P1           │
  │ O23 0 0 1   │
  │ O31 0 1 0   │
  ├─────────────┤
  │ ..           │
  │ ..           │
  │ ..           │
  │ ..           │
  └─────────────┘

  ┌─────────────┐
  │ M1 M2 M3    │
  ├─────────────┤
  │ O11 0 0 1   │
  │ O21 1 0 0   │
  ├─────────────┤
  │ O22 0 1 0   │
  │ O12 0 0 1   │
  │ O13 1 0 0   │
  ├─────────────┤
  │ P2           │
  │ O23 0 0 1   │
  │ O31 0 1 0   │
  ├─────────────┤
  │ ..           │
  │ ..           │
  │ ..           │
  │ ..           │
  └─────────────┘

  ➢ This is performed for certain iteration (in this paper, 30 iteration)
7) Environmental selection

- Chromosomes must be sorted based on their HIS in ascending order (in minimization problem). The new population is formed with the N best solutions.

8) Stopping criterion

- 150 iteration or a maximum time limit of \( (n \times n_t \times m \times 0.1s) \) of CPU time
- No improvement : 20
4. Computational experiment

- **Experiment environment**

  - Two kinds of objective function: makespan and Aggregate objective function (AOF)
  
  - Two problems:
    
    
    b. artificial benchmarks according to the function proposed by Bagheri and Zandieh (2011)

  - Comparing proposed HGA against the available algorithm in literature
    
    a. Variable neighbourhood search (VNS) from Bagheri and Zandieh (2011)
    
    b. Tabu search (TS) from Ennigrou and Ghedira (2008)
    
    c. Artificial Immune System (AIS) and Particle swarm Optimization (PSO) from Sadrzadeh (2013)

  - JAVA and run on PC with core2Duo 2.6 GHZ and 2GB RAM

  - The following values are more effective for the two problems

    - The population size \( (N) \): 150
    - Crossover rate = 0.6
    - Mutation rate = 0.2
    - Local search rate = 0.2
    - Number of iteration of HGA (stopping condition) = 150 or CPU time limit fixed to \( n \times n_i \times m \times 0.1 \) s
    - Number of no improvement (stopping condition) = 20
    - Number of iteration of VNS: 30
3. Solution approach

4. Computational experiment

* Computational result for the makespan

<table>
<thead>
<tr>
<th>Instance problem</th>
<th>Size n×m</th>
<th>VNS</th>
<th>TS</th>
<th>GA</th>
<th>HGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>La01</td>
<td></td>
<td>2.54</td>
<td>7.36</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>La02</td>
<td></td>
<td>1.58</td>
<td>8.53</td>
<td>2.57</td>
<td>0.33</td>
</tr>
<tr>
<td>La03</td>
<td>10×5</td>
<td>2.66</td>
<td>5.33</td>
<td>0.13</td>
<td>0.54</td>
</tr>
<tr>
<td>La04</td>
<td></td>
<td>3.50</td>
<td>3.66</td>
<td>4.07</td>
<td>0.90</td>
</tr>
<tr>
<td>La05</td>
<td></td>
<td>1.11</td>
<td>3.58</td>
<td>0.16</td>
<td>0.24</td>
</tr>
<tr>
<td>La06</td>
<td></td>
<td>4.45</td>
<td>2.79</td>
<td>2.04</td>
<td>0.32</td>
</tr>
<tr>
<td>La07</td>
<td></td>
<td>2.92</td>
<td>2.99</td>
<td>1.97</td>
<td>0.34</td>
</tr>
<tr>
<td>La08</td>
<td>15×5</td>
<td>2.37</td>
<td>2.41</td>
<td>2.86</td>
<td>0.65</td>
</tr>
<tr>
<td>La09</td>
<td></td>
<td>3.82</td>
<td>3.21</td>
<td>0.41</td>
<td>0.33</td>
</tr>
<tr>
<td>La10</td>
<td></td>
<td>1.84</td>
<td>4.02</td>
<td>0.00</td>
<td>0.62</td>
</tr>
<tr>
<td>La11</td>
<td></td>
<td>3.32</td>
<td>2.08</td>
<td>1.61</td>
<td>0.34</td>
</tr>
<tr>
<td>La12</td>
<td></td>
<td>5.33</td>
<td>2.71</td>
<td>2.38</td>
<td>0.60</td>
</tr>
<tr>
<td>La13</td>
<td>20×5</td>
<td>3.30</td>
<td>3.14</td>
<td>3.02</td>
<td>1.02</td>
</tr>
<tr>
<td>La14</td>
<td></td>
<td>1.70</td>
<td>1.76</td>
<td>1.26</td>
<td>0.93</td>
</tr>
<tr>
<td>La15</td>
<td></td>
<td>2.37</td>
<td>1.18</td>
<td>1.97</td>
<td>1.95</td>
</tr>
<tr>
<td>La16</td>
<td></td>
<td>2.37</td>
<td>4.04</td>
<td>0.00</td>
<td>0.76</td>
</tr>
<tr>
<td>La17</td>
<td></td>
<td>4.34</td>
<td>2.39</td>
<td>0.07</td>
<td>0.14</td>
</tr>
<tr>
<td>La18</td>
<td>10×10</td>
<td>1.86</td>
<td>5.81</td>
<td>0.91</td>
<td>0.35</td>
</tr>
<tr>
<td>La19</td>
<td></td>
<td>6.37</td>
<td>6.67</td>
<td>0.11</td>
<td>0.17</td>
</tr>
<tr>
<td>La20</td>
<td></td>
<td>4.84</td>
<td>9.36</td>
<td>2.48</td>
<td>0.35</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>3.12</td>
<td>4.15</td>
<td>1.40</td>
<td>0.54</td>
</tr>
</tbody>
</table>

- 20 instances derived from Hurikn et al (1994)
- Operations per jobs is equal to the number of machines
- The same setup time matrix was added for each machine in all benchmark instances
- Ten runs of each generated instance, the best solution obtained for each instance are calculated
- Relative percentage deviation (RPD) measure

\[
\text{RPD} = \frac{\text{Sol(algo)} - \text{Sol(min)}}{\text{Sol(min)}} \times 100
\]
3. Solution approach

4. Computational experiment

- The proposed HGA performs better than the others algorithms in 13 instances
- The proposed HGA outperforms the others algorithms with average RPD of 0.54
- The Proposed HGA obtained the best average RPD of 0.35 in the largest number of machine
- The proposed HGA keeps its robust performance in different problem size

### Computational result for the makespan

**Table 3. Summary of results in the SDST-FJSP to minimize the makespan: SDST-HU data benchmark.**

<table>
<thead>
<tr>
<th>Instance</th>
<th>Size</th>
<th>VNS</th>
<th>TS</th>
<th>GA</th>
<th>HGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>La01</td>
<td>2.54</td>
<td>7.36</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>La02</td>
<td>1.58</td>
<td>8.53</td>
<td>2.57</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>La03</td>
<td>2.66</td>
<td>5.33</td>
<td>0.13</td>
<td>0.54</td>
<td></td>
</tr>
<tr>
<td>La04</td>
<td>3.50</td>
<td>3.66</td>
<td>4.07</td>
<td>0.90</td>
<td></td>
</tr>
<tr>
<td>La05</td>
<td>1.11</td>
<td>3.58</td>
<td>0.16</td>
<td>0.24</td>
<td></td>
</tr>
<tr>
<td>La06</td>
<td>4.45</td>
<td>2.79</td>
<td>2.04</td>
<td>0.32</td>
<td></td>
</tr>
<tr>
<td>La07</td>
<td>2.92</td>
<td>2.99</td>
<td>1.97</td>
<td>0.34</td>
<td></td>
</tr>
<tr>
<td>La08</td>
<td>2.37</td>
<td>2.41</td>
<td>2.86</td>
<td>0.65</td>
<td></td>
</tr>
<tr>
<td>La09</td>
<td>3.82</td>
<td>3.21</td>
<td>0.41</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>La10</td>
<td>1.84</td>
<td>4.02</td>
<td>0.00</td>
<td>0.62</td>
<td></td>
</tr>
<tr>
<td>La11</td>
<td>3.32</td>
<td>2.08</td>
<td>1.61</td>
<td>0.34</td>
<td></td>
</tr>
<tr>
<td>La12</td>
<td>5.33</td>
<td>2.71</td>
<td>2.38</td>
<td>0.60</td>
<td></td>
</tr>
<tr>
<td>La13</td>
<td>3.30</td>
<td>3.14</td>
<td>3.02</td>
<td>1.02</td>
<td></td>
</tr>
<tr>
<td>La14</td>
<td>1.70</td>
<td>1.76</td>
<td>1.26</td>
<td>0.93</td>
<td></td>
</tr>
<tr>
<td>La15</td>
<td>2.37</td>
<td>1.18</td>
<td>1.97</td>
<td>1.95</td>
<td></td>
</tr>
<tr>
<td>La16</td>
<td>2.37</td>
<td>4.04</td>
<td>0.00</td>
<td>0.76</td>
<td></td>
</tr>
<tr>
<td>La17</td>
<td>4.34</td>
<td>2.39</td>
<td>0.07</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>La18</td>
<td>1.86</td>
<td>5.81</td>
<td>0.91</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>La19</td>
<td>6.37</td>
<td>6.67</td>
<td>0.11</td>
<td>0.17</td>
<td></td>
</tr>
<tr>
<td>La20</td>
<td>4.84</td>
<td>9.36</td>
<td>2.48</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>3.12</td>
<td>4.15</td>
<td>1.40</td>
<td>0.54</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.** The average RPD of the algorithms versus the number of jobs.
4. Computational experiment

- Computational results for the AOF

  Artificial benchmarks according to the function proposed by Bagheri and Zandieh (2011)

<table>
<thead>
<tr>
<th>Class</th>
<th>n×m×m</th>
<th>AMO</th>
<th>Processing time</th>
<th>SDST</th>
<th>Dummy Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class1</td>
<td>10×5×5</td>
<td>U(1,5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class2</td>
<td>15×5×8</td>
<td>U(1,8)</td>
<td>U(20,100)</td>
<td>U(20,60)</td>
<td>U(20,40)</td>
</tr>
<tr>
<td>Class3</td>
<td>10×10×5</td>
<td>U(1,5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class4</td>
<td>15×10×10</td>
<td>U(1,10)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Due dates are generated using this formula as in Bagheri and Zandieh (2011)

\[
d_i \sim U \left[ \mu_i \times \left(1 - \frac{R}{2}\right); \mu_i \times \left(1 + \frac{R}{2}\right) \right]
\]

\[
\mu_i = \left(1 + \frac{T \times n}{m}\right) \times \sum_{j=1}^{n_i} \text{PS}_{i,j}
\]

\(T=0.3, \ R=0.5\)

\(i=\) job index, \(j=\) operation index, \(m=\) # of machine, \(n=\) # of job

\(n_i=\) # of operations of job \(i\)

\(\text{PS} = \) average of total setup and processing times for each operation
Computational results for the AOF

- Overall, compared to VNS, AIS, PSO and GA, proposed HGA has a superiority result to minimize the AFO for all $\alpha$ values.

- HGA is more effective with $\alpha = 0.25$ then $\alpha = 0.75$. Otherwise, The HGA have the best results with mean tardiness against makespan objective function.

**Figure 4.** The average RPD of the algorithms of each type of problem class for $\alpha = 0.25$.

**Figure 5.** The average RPD of the algorithms of each type of problem class for $\alpha = 0.5$.

**Figure 6.** The average RPD of the algorithms of each type of problem class for $\alpha = 0.75$. 
5. Conclusion

- **flexible job shop scheduling problem with sequence dependent setup times**
  - Hybrid genetic algorithm is proposed to minimize makespan and aggregate objectives function
  - To compare the performance of HGA, two kinds of benchmark are done
  - Results shows that the proposed HGA is better than other algorithms
  - In future works, it will investigate the dynamic scheduling problem to closely reflect the real flexible job shop scheduling environment

- **Adv & Disadv**
Thank you