A Tabu-Search Based Constructive Hyper-heuristics for Scheduling Problems in Textile Industry

Cecilia E. Nugraheni and Luciana Abednego
Dept. of Informatics, Parahyangan Catholic University, Bandung, Indonesia
Email: cheni@unpar.ac.id

Abstract—One type of scheduling problems found in the area of textile industry is the Flow Shop Scheduling Problem (FSSP). The solution of this problem is the sequence of jobs that meet certain requirements. In practice, dispatching rules are often used to determine the order of jobs. This paper proposes a new approach in finding solutions for FSSP. The proposed method applies a tabu-search based hyper-heuristic approach. Hyper-heuristic can be used to combine several basic heuristics to obtain a solution. By using four dispatching rules as the basic heuristics, i.e. FIFO, LIFO, SPT, and LPT, the proposed method, TSHH, has been implemented in a computer program. In order to measure the performance of TSHH, several experiments have been conducted by using Taillard's benchmark datasets. The conclusion obtained is that with a proper heuristics combination and value tenure, TSHH is able to produce better solutions than the solutions of four dispatching rules.

Index Terms—flow shop scheduling problem, dispatching rules, tabu-search, hyper-heuristics

I. INTRODUCTION

One of the important industries in Indonesia is the textile industry. The textile subsector plays as an important source of foreign exchange. Besides, it is also a labor-intensive industry that absorbs a lot of labor. Nowadays, this industry suffers from some problems. One of them is the condition of production machines. Textile production machines owned by textile factories can be categorized as old machines (around 20 years old). The use of old production machines may cause a large amount of energy consumption and may affect the working speed optimality and the product quality as well. One of the possible solutions of this problem is a good production scheduling. As a manufacturing industry, the optimal production scheduling becomes a very important issue.

Scheduling is understood as assigning jobs to machines or operators for specified time period satisfying some constraints. In the context of textile industry, in general, there are two types of scheduling problems: Flow Shop Scheduling Problem (FSSP) and Job Shop Scheduling Problem (JSSP). This work is focused on FSSP. Given \( m \) machines and \( n \) jobs that will be processed on each machine, an FSSP is a problem to find a sequence of jobs that satisfies some particular criteria. One of the important objectives is minimum makespan, which is the time between the beginning of the execution of the first job on the first machine and the completion of the execution of the last job of the sequence on the last machine.

There are many work dedicated to FSSP. The proposed methods, in general, are classified into two techniques: constructive or improvement [1], [2]. The two techniques differ in the way they construct the job sequence. A constructive technique iteratively builds the sequence by adding jobs one by one into the sequence, whereas an improvement technique starts with an initial solution and then by using particular mechanism it iteratively changes the sequence in order to find the best solution [3].

In practice, frequently people apply some heuristics called dispatching rules for constructing the job ordering. A dispatching rule is used to select one of the waiting jobs with the highest priority to be processed. There are many dispatching rules. Four basic dispatching rules considered in this work are FIFO (First in First Out), LIFO (Last In First Out), SPT (Shortest Processing Time), and LPT (Longest Processing Time).

The weaknesses of each approach are the lack of generality. It performs well only for a specific problem instance. Hyper-heuristics are proposed to overcome this problem. Hyper-heuristics are search methodologies for choosing or generating (combining, adapting) heuristics (or components of heuristics), in order to solve a range of
optimization problems [4]. Rather than search a space of solution directly, hyper-heuristics search a space of heuristics. Fig. 1 shows the general framework for the hyper-heuristics approach.

A survey on heuristics for scheduling problem in textile industry is given in [5]. There are nine heuristics that can be used for solving FSSP. In [6] we have proposed a genetic programming based hyper-heuristics approach for combining two of nine heuristics reported in [5].

From AI point of view, FSSP can be regarded as a searching problem, i.e. finding a solution in a solution space. There are many popular searching algorithms. One of them is tabu search. The advantage of tabu search compared to other searching algorithms is its ability to avoid the local optima. This is done by using a tabu list for memorizing which solutions or candidate solutions has already visited.

In this work, we investigate the combination of hyper-heuristics and tabu search for solving FSSP. Many works similar to our work can be found in the literature. Among them, the work that is most closely related to our work is the work from Graham and Hussin [7]. In [7], they presented a tabu search hyper-heuristics for solving timetabling problems. The contribution of our work is the combining four heuristics for solving examination timetabling problems. As far as our knowledge, this topic is not yet reported in the literature.

The rest of this paper is structured as follows. Section 2 gives a brief explanation of FSSP. Section 3 explains the proposed algorithm. Section 4 presents the experimental results. Last, conclusion and future work are given in Section 5.

II. FLOW SHOP SCHEDULING PROBLEMS

Following [3], flow shop scheduling problem can be defined as follows: *Given n jobs (items, tasks, ...) to be processed in the same sequence on m machines; the processing time of job i on machine j is fixed and given by t_{ij} (t_{ij} > 0). The flow shop scheduling problem consists of minimizing the time between the beginning of the execution of the first job on the first machine and the completion of the execution of the last job on the last machine; this time is called makespan.*

There are some assumptions made for this problem:

- Every job has to be processed at most once on machine 1, 2, ..., m.
- Every machine processes only one job at a time
- Every job is processed at most on one machine at a time.
- The operations are not preemptable.
- The set-up times of the operations are included in the processing time and do not depend on the sequence.
- The operating sequences of the jobs are the same on every machine and the common sequence has to be determined.

As an illustration, let's consider a small FSSP. There are five jobs that will be processed on three machines. The processing times required for every job by every machine are given in Table I. Schedule resulted by applying dispatching rule FIFO, LIFO, SPT, and LPT is (j_1, j_2, j_3, j_4, j_5), (j_5, j_4, j_3, j_2, j_1), (j_5, j_2, j_3, j_4, j_1), and (j_1, j_3, j_5, j_4, j_2), respectively.

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Machines 1</th>
<th>Machines 2</th>
<th>Machines 3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>j_1</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>j_2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>j_3</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>j_4</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>j_5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

Assume we apply LIFO for constructing the schedule, the resulted makespan is 22.

III. PROPOSED ALGORITHM

Given a set of n jobs, a set of k heuristics, and a set of k job ordering associated to each heuristic, the algorithm of the proposed method, TSHH, is as follows. Initially all heuristics are non-tabu heuristics (t_{ih} = 0). Starts with initial solution candidate, which is an empty job ordering. TSHH constructs an n-length job ordering by selecting a non-tabu heuristic iteratively. Whenever a non-tabu heuristic h is found, the job with the highest priority according to heuristic h is inserted to the solution candidate. Every time a non-tabu heuristic h is selected, it becomes tabu heuristic and remains tabu until its tabu tenure value t_{ih} equals to 0.

In algorithmic notation, TSHH algorithm is given by Fig. 2.

```plaintext
let J = \{j_1, ..., j_n\} be a set of n jobs, H = \{h_1, ..., h_k\} be a set of k heuristics, and O = \{o_{i_1}, ..., o_{i_k}\} be a set of job ordering o_i associated to h_i on J.

for all i \in \{1, ..., n\}, t_i \leftarrow 0
i \leftarrow 1
while i < n do
    let h_m \in J be some heuristic in H
    if t_m = 0 then
        t_m \leftarrow t
        s \leftarrow o_{o_{m}(s)}/* the first job of ordering o_m */
        J \leftarrow J \setminus \{j_m\}
    for all r \in \{1, ..., k\}: delete j_m from o_r
    for all r \in \{1, ..., k\} s.t. r \neq m and t_r > 0:
        t_r \leftarrow t_r - 1
        i \leftarrow i + 1
    else
        for all r \in \{1, ..., k\}: t_r \leftarrow t_r - 1
endif
endwhile

⇒ s/* the resulted job ordering */
```
IV. COMPUTATIONAL EXPERIMENTS

A. Experiment Setup

For measuring the performance of the proposed method, some experiments have been conducted. The objective of the experiments is to answer the following questions:

1. How is the performance of the proposed method, TSHH, compared to the original dispatching rules?
2. Which combination of dispatching rules performs the best?
3. Does the tabu tenure have an impact on the performance of TSHH?

In this work, we ran our experiments with the objective of minimizing the makespan of Taillard’s benchmark problem datasets [9]. Taillard’s benchmark consists of 8120 instances, 10 each of one particular size. Taillard’s datasets range from 20 to 500 jobs and 5 to 20 machines. For comparison purposes, following [9], we use the outputs of NEH algorithm as reference solutions.

For every problem instance, we ran each dispatching rule (FIFO, LIFO, SPT, and LPT) and the proposed method, TSHH, with eleven combination of dispatching rules (FIFO-LIFO-SPT-LPT, FIFO-LIFO-SPT-LPT, FIFO-LIFO-LPT-LPT, FIFO-LIFO, FIFO-LIFO-LPT, FIFO-LIFO-LPT, SPT-LPT-LPT) and seven different tabu tenure values (0.1, 2, 3, 4, 5, and 10). Thus, for each problem instance, we have produced 81 makespans.

For each problem instance, we calculate the minimum, maximum, and median values based on 81 resulted makespans. The minimum and maximum values will be used to know the best and the worst performance of TSHH, whereas the median value will be used as indicator of TSHH performance in general. The used of median value instead of mean (average) in stochastic algorithmic performance is inspired by Ikkovic et.al. in [10].

For comparing the solutions of each dispatching rule and the proposed algorithm, following [8], we used Relative Percent Deviation (RPD) and Average Relative Percent Deviation (ARPD). The definition of Relative Percent Deviation and Average Percentage Relative Deviation is respectively given by:

\[
\text{RPD}_i = \frac{HS_i - RS_i}{RS_i} \times 100\% \quad (1)
\]

\[
\text{ARPD} = \frac{1}{I} \sum_{i=1}^{I} \text{RPD}_i \quad (2)
\]

where:

- \(I\) number of problem instances,
- \(HS_i\) heuristic solution of problem instance \(i\),
- \(RS_i\) reference solution of problem instance \(i\),
- \(\text{RPD}_i\) percentage relative deviation of problem instance \(i\).

B. Experimental Results

Table II shows the makespans for all problem instances and for all datasets resulted by each dispatching rule and the median, the best, and the worst makespans produced by TSHH. The median, the best, and the worst makespan resulted by TSHH is denoted by M-TH, B-TH, and W-TH, respectively. PI stands for Problem Instance and RMB stands for Referenced Benchmark.

### Table II: Resulted Makespans

<table>
<thead>
<tr>
<th>Dataset 20 jobs 5 machines</th>
<th>PI</th>
<th>RMB</th>
<th>FIFO</th>
<th>LIFO</th>
<th>SPT</th>
<th>LPT</th>
<th>M-TH</th>
<th>B-TH</th>
<th>W-TH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 20 jobs 10 machines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dataset 50 jobs 5 machines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dataset 50 jobs 10 machines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dataset 50 jobs 20 machines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2017 Journal of Industrial and Intelligent Information 25
Dataset 200 jobs 10 machines
1 5514 5943 6260 5222 5980 6182 5976 6419
2 5264 5878 5988 5834 5759 5757 5712 6044
3 5222 5880 5819 6006 5968 5939 5647 6082
4 5023 5675 5656 5731 5006 5667 5412 5863
5 5261 6095 6067 5868 5934 6020 5867 6313
6 5154 5753 5789 5832 5743 5717 5539 5923
7 5282 5935 6029 6082 6059 6037 5779 6351
8 5140 6068 5738 6032 5748 5931 5612 6148
9 5489 6193 6082 6211 6152 6215 5924 6363
10 5336 6157 5897 6140 5886 6115 5838 6556

Dataset 100 jobs 5 machines
1 5897 6982 6742 6805 6823 6848 6608 7767
2 5466 6558 6504 6214 6677 6523 6182 7727
3 5747 6667 6635 6886 6831 6828 6596 8072
4 5924 7300 7052 6774 6919 6968 6692 7981
5 5672 6844 6417 6788 6505 6699 6428 7766
6 5395 6591 6319 6394 6470 6495 6282 7848
7 5717 6765 6545 6651 6730 6639 6407 7905
8 5752 6517 6761 6735 6611 6698 6407 8223
9 6016 6859 6941 6962 6712 6930 6767 7817
10 5937 6693 6915 6462 6697 6752 6491 8282

Dataset 100 jobs 10 machines
1 6520 7840 7846 7700 7586 7766 7562 8016
2 6550 7591 7690 7783 7646 7744 7514 7925
3 6621 7755 7915 7719 7667 7958 7633 8370
4 6589 7885 7777 7698 7467 7855 7537 8172
5 6697 7729 7808 8021 7828 7765 7562 8036
6 6813 8072 7849 7770 8085 7947 7734 8184
7 6578 8033 7992 7851 7598 7906 7715 8129
8 6791 8138 8204 8309 8113 8220 7894 8368
9 6679 7907 7880 7711 7938 7830 7701 8155
10 6680 8099 7936 8206 7866 8073 7661 8512

Resulted ARPD of each dispatching rules combination.

From the resulted makespans in Table II, we have calculated the RPD and the ARPD of each heuristic. The ARPD of FIFO, LIFO, SPT, LPT, M-TH, B-TH, and W-TH, are shown in Fig. 3. From Fig. 3, it can be concluded that TSHTH tends to produce results that are not much different from the results from original dispatching rules. This is based on the value of M-TH. However, for some cases, TSHTH can yield a much better, and even worse, solution.

To find out which combination of dispatching rules is the best, we have calculated the ARPD relatively to each possible combination. Since there are four dispatching rules, there are eleven possible combinations to consider. The ARPDs of all dispatching rule combinations are given by Fig. 4. We refer FIFO, LIFO, SPT, and LPT by number 1, 2, 3, and 4, respectively. For examples, H-134 represents the combinations of FIFO, SPT, and LPT, and H-23 represents the combinations of LIFO and SPT.
Fig. 4 shows the resulted ARPD of each dispatching rules combination. It is clear that FIFO-LPT produced the best result.

Last, we have calculated the ARPD of each tabu tenure value. The resulted computation is given by Fig. 5. From Fig. 5, in average, we can see that tabu tenure 5 yields the best result.

V. CONCLUSIONS AND FUTURE WORK

We have presented an algorithm for finding the solution of Flow Shop Scheduling Problem using tabu search based hyper-heuristics approach. From the experimental results, we have shown that this approach is capable of producing a better solution than the solutions resulted in the original dispatching rules. It is also shown that there is no guarantee that this approach will always give the best solution. However, with a proper selection of dispatching rules combination and tabu tenure settings, the proposed method will perform well. The best dispatching rules combination is FIFO-LPT, whereas the best tabu tenure value is 5.

As future work, we will continue our work on the other type of scheduling problem found in textile industry which is Job Shop Scheduling Problem. We will investigate whether this approach can be applied to this scheduling problem and measure the performance. Besides makespan, some other optimization criteria will also be considered.

ACKNOWLEDGMENT

This work was supported in by the Ministries of research, technology, and higher education of Republic Indonesia under Grant DIPA-042.06.1.401516/2017.

REFERENCES


Cecilia E. Nugraheni received the bachelor and master degrees in Informatics Engineering from Institut Teknologi Bandung, Indonesia. She received the Ph.D. degree in Informatics from Institut für Informatik, Ludwig-Maximilians Universität, Munich, Germany. Currently, she is a lecturer at Informatics Dept. Parahyangan Catholic University, Bandung, Indonesia. Her research interests are formal specification and verification of reactive systems, multi agent systems, soft computing, production scheduling optimization, meta- and hyper-heuristics.

Luciana Abednego received the bachelor degree in Computer Science from Parahyangan Catholic University. She received the master degree in Informatics Engineering from Institut Teknologi Bandung. Currently, she is a lecturer at Informatics Dept. Parahyangan Catholic University, Bandung, Indonesia. Her research interests include programming and algorithms, computer graphics, machine learning, production scheduling optimization, meta- and hyper-heuristics.