MINIMIZATION OF THE MEAN ABSOLUTE DEVIATION FROM A COMMON DUE DATE IN A TWO-MACHINE FLOWSHOP

Débora P. Ronconi\textsuperscript{a}, Celso S. Sakuraba\textsuperscript{a} and Francis Sourd\textsuperscript{b}

\textsuperscript{a}Departamento de Engenharia de Produção, EPUSP, Universidade de São Paulo, Av. Prof. Almeida Prado, 128, Cidade Universitária, 05508-900, São Paulo SP, Brazil

\textsuperscript{b}Lab. LIP6, Université Pierre et Marie Curie, Case Courier 169, 4, Place Jussieu, 75252 Paris, Cedex 05, France

Abstract_ This paper addresses the minimization of the mean absolute deviation from a common due date in a two-machine flowshop scheduling problem. Initially, a job scheduling algorithm that obtains an optimal schedule for a given job sequence is presented. This algorithm is used with a job insertion procedure to generate a group of heuristics that differ on the initial job sequencing rule. Computational experiments show that the developed heuristics outperform results found in the literature for problems up to 500 jobs.

Keywords: scheduling, earliness and tardiness, common due date, flowshop, heuristics.

1. Introduction

This paper addresses the flowshop scheduling problem. This environment is characterized by \( n \) jobs being processed on \( m \) machines always in the same order, that is, the \( k \)-th operation of every job must be conducted on machine \( k \). We will consider the case with two machines (\( m = 2 \)) and with the same job sequence in both machines, known as permutation schedule. Using that assumption, the number of possible sequences is reduced from \((n!)^2\) to \(n!)\). According to Kim (1995), permutation schedules do not always include the optimal schedule for a given problem, but their importance should not be underestimated, as in most real situations, only permutation schedules are feasible. A static and deterministic environment is assumed here, where the processing times and due dates are given and all jobs are available for processing since the beginning. The objective function is the minimization of the mean absolute deviation (MAD) of job completion times from a common due date.

Meeting due dates is a common objective for many manufacturing processes. Tardy jobs may generate contractual penalties and loss of credibility, causing damages to the company’s image and loss of clients (Sen and Gupta, 1984). Early jobs were discouraged since the advent of Just-in-Time approaches due to the costs generated by those jobs, such as tied-up capital and inventory costs (Biskup and Feldmann, 2001).

A particular case of MAD minimization occurs when all jobs have the same due date. Such scenarios can be found, for example, at a chemical company where different chemicals are manufactured using the same process and have to be combined as close as possible to a common due date to avoid deterioration (Sarper, 1995). A wide variety of studies involving scheduling problems with earliness and tardiness penalties and common due date can be found in the literature, such as those described in the reviews presented by Baker and Scudder (1990) and Gordon \textit{et al.} (2002). Nonetheless, both papers only addressed scenarios with single or parallel machines.

* Corresponding author: Tel.: +55-11-30915363, Fax: + 55-11-3091 5399.
E-mail address: dronconi@usp.br
For the single machine problem, Kanet (1981) proposes an algorithm that finds an optimal solution in polynomial time when the due date is greater than the sum of processing times of the jobs. This algorithm lists jobs in non-increasing order of processing times and schedules them as follows: jobs in odd positions are scheduled in the same order without idle time, so that the last job of the sequence completes its processing exactly on the due date; jobs in even positions are scheduled in reverse order, with no idle time either and with the first job starting its processing exactly on the due date. Bagchi et al. (1986) extend the results of Kanet and present an algorithm, which determines multiple optimal schedules when the due date is greater than $\Delta$ (where $\Delta$ is the sum of processing times of the jobs in odd positions if $n$ is odd, or of jobs in even positions if $n$ is even; it is assumed that jobs are in non-decreasing order of processing times). The authors also present some properties and an implicit enumeration procedure for problems with due dates smaller than $\Delta$.

Sung and Min (2001) present a study on minimizing MAD from a common due date in a flowshop scenario. Those authors deal with a two-stage flowshop where there are batch processing machines (BPM) that can process a given number of jobs simultaneously. Properties for three cases of problems involving BPM are presented, dealing not only with the job sequencing on both machines, but also with the efficient batch formation criteria. Yeung et al. (2004) deal with a two-machine flowshop aiming to minimize the sum of absolute deviations from a due window instead of a due date. They present several properties of the problem and develop a heuristic based on these properties. Right-shifting and swapping of jobs are used in order to obtain a schedule that satisfies the properties. Pairwise interchange is also performed to improve the solution. This heuristic is utilized as an initial upper bound for a branch-and-bound method. Furthermore, lower bounds are proposed and the complexity of four special cases is analyzed.

As far as we know, only Sarper (1995) addressed the common due date MAD minimization in a two-machine flowshop problem. Sarper proposed a Mixed Integer Linear Programming (MILP) model for the nonpermutation scheduling problem and a job scheduling algorithm for a given job sequence. This algorithm is based on the author’s assumption that it is not necessary to have idle time between jobs on the first machine, and can be briefly described as follows: in the first step, jobs are scheduled to start as early as possible on both machines; then, one time unit is added to the starting time of all jobs on the second machine. The last step is repeated until there is no improvement on the objective function. The scheduling algorithm is used to evaluate the sequence together with three heuristics that construct the sequences, generating three constructive heuristics, namely H1, H2 and H3.

According to Sarper (1995), H3 heuristic provides the best performance using a sequence construction procedure similar to the NEH algorithm (Nawaz et al., 1983). In this method, the jobs are first sorted using their processing times on the second machine in LPT (Longest Processing Time) order. Then, jobs are inserted one by one following this initial list in all possible positions of the previous sequence. The first job is inserted in the first position. The second job is inserted before and after the first job, keeping the best sequence. The third job is inserted in the beginning, between the first two jobs, and in the end, once again keeping the best sequence, and so on until all jobs are inserted. The job scheduling algorithm proposed by the author is used to evaluate partial sequences. After obtaining the final sequence, the binary variables that indicate the sequence on the MILP model are fixed and this model is used to determine the optimal schedule. Comparing the results with the optimal values for problems up to 6 jobs, H3 has a mean deviation under 3%. That performance is not as good as H1, but Sarper claims that the performance of H3 is superior for larger problems (from 10 to 20 jobs). Furthermore, H2 and H3 have a shorter CPU time than H1.

Since there is a small number of papers that deal with MAD minimization for the two-stage flowshop problem, this study presents heuristics that apply a proposed algorithm to find an optimal schedule for a given sequence. These heuristics showed improvement when compared with the results found in the literature for the restricted case. A common due date is called
unrestrictive if its optimal value has to be determined or if it is given and has no influence on the optimal schedule (Felmann and Biskup, 2003). The single-machine with restricted due date problem is NP-Hard (Hall et al., 1991), and so is the two-machine case, which justifies the choice of heuristic methods for its solution.

This paper is organized as follows: the next section describes the proposed scheduling algorithm, while Section 3 presents the results of using this algorithm associated with sequencing heuristics. The last section summarizes the main results.

2. Timing Algorithm

The heuristics proposed in this paper include two stages: first, the job sequence is constructed and then a schedule is defined. For the construction of the sequences, we use procedures based on the NEH heuristic proposed by Nawaz et al. (1983), while for the schedule we present an algorithm that provides the optimal schedule for a given job sequence. More precisely, this algorithm computes the start times for each operation and is called “Timing Algorithm” in reference to similar algorithms for the earliness-tardiness problem with distinct due dates (Hendel and Sourd, 2007).

This algorithm is based on the resolution of an equivalent problem. As there is no cost on operations on the first machine, these operations can be scheduled as soon as possible. Then, the completion time of a job on the first machine can be seen as the ready time for the second one. Therefore, the two-machine flowshop timing problem reduces to a one machine timing problem with release dates and common due date. To the best of our knowledge, this problem has not been studied in the literature.

In the following, we prove that the timing problem associated to \( I | r, d_i = d | \sum \alpha_i E_i + \beta_i T_i \) can be solved in \( O(n) \) time. Jobs are assumed to be indexed in their sequence order. Each job has a duration \( p_i \), a release date \( r_i \), an earliness penalty \( \alpha_i \) and a tardiness penalty \( \beta_i \) (in the flowshop problem we have \( \alpha_i = \beta_i = 1 \)). The validity of this algorithm derives from properties of timing algorithms with convex cost functions (see Chrétienne and Sourd (2003)). We however give sufficient details for the understanding of the algorithm and the proof of its complexity.

We first observe that there is no idle time after an early job \( i \), otherwise, this job can be postponed until its completion time \( C_i \) reaches the start time of job \( i+1 \) or the common due date \( d \). Similarly, after \( d \), any inserted idle time is costly, which means that, if there is some idle time before a tardy job \( i \), this job necessarily starts at its release date \( r_i \). Therefore, there is an optimal schedule such that the first \( q \) tasks are scheduled in a first block and the last \( n-q \) last task are scheduled at earliest after \( r_k \).

Let \( S \) be the start time of the first block and let \( C = S + P_q \) (with \( P_q = \sum_{i=1}^q p_i \)) be its completion time. The cost for scheduling this block is given by \( F_q(S) = \sum_{i=1}^q f_i(S + p_i) \) with \( f_i(t) = \max\{\alpha_i(d-t), \beta_i(t-d)\} \). Since each \( f_i \) is convex piecewise linear, \( F_q \) is convex piecewise linear. Therefore, a simple analysis of \( F_q \) shows that \( F_q \) reaches its minimum for \( S = d - P_{h_{i-1}} \) where \( h_i \) is the smallest index such that \(-\sum_{i=1}^{h_i-1} \alpha_i + \sum_{i=h_i}^{q} \beta_i < 0 \) (note that this expression is non-increasing with \( h \) and represents the left derivative of \( F_q \) for \( S = d - P_{h_{i-1}} \)). Observe that \( h \) is the index of the first late job in the schedule. Moreover, the release date constraints must be satisfied, that is \( S + P_{i-1} \geq r_i \) for \( i = 1 \ldots q \), which means that the optimal start time for scheduling in a single block the first \( q \) tasks is \( S = \max\{d - P_{h_{i-1}}, r_i - P_{i-1} | i = 1 \ldots q \} \).
The timing algorithm iteratively computes $S$ for the problem restricted to the first $j$ tasks and it stops as soon as it can be detected that scheduling the remaining tasks will not modify the starting time $S$, which is guaranteed as soon as a task in $1,\ldots, j$ must start at its release date. From the above analysis, we observe that the value of $S$ computed for the first $j-1$ tasks cannot be greater than the value of $S$ computed for the first $j$ tasks, which means that $S$ can only be decreased when the $j^{th}$ task is added to the schedule.

At the end of step $j-1$, $S$, $C$ and $h$ are assumed to be up-to-date. We also maintain $\alpha = \sum_{i=1}^{j} \alpha_i$ and $\beta = \sum_{i=1}^{j} \beta_i$. The expression $\beta - \alpha$ represents the left derivative of $F_{j-1}$ at value $S$ denoted by $F'_{j-1}(S^-)$. When task $j$ is added to the schedule, we first check whether $r_j \geq C$ which means that all the remaining jobs have to be scheduled after $r_j$ and the first block is optimally scheduled. Otherwise, we add $j$ to the block and update $\beta$. If $F'_{j-1}(S^-) = \beta - \alpha$ becomes positive, $S$ has to be decreased to minimize $F_j$, that is $h$ must be increased or, in other words, the current job $h$ becomes on-time. Variable $\Delta$ represents the minimum amount of time between the start of a job and its release date, that is $\Delta = \min\{S + P_i - r_i \mid 1 \leq i \leq j\}$. When $S$ is decreased by $\delta$, $\Delta$ is also decreased by $\delta$ and is therefore updated in $O(1)$ time. By definition, $S$ cannot be decreased by a value greater than $\Delta$ without violating a release date. Therefore, if $\Delta$ becomes null, it means that a task in the block starts at its release date and, therefore, $S$ cannot be decreased by the insertion of the remaining tasks in the schedule.
Algorithm 1 Timing algorithm

\[
S = d; \quad C = d
\]
\[
\alpha = 0; \quad \beta = 0
\]
\[
h = 1
\]
\[
\Delta = d
\]
\[
j = 1
\]
while \( j \leq n \), \( \Delta > 0 \) and \( r_j < C \) do
\[
C = C + p_j
\]
\[
\beta = \beta + \beta_j
\]
\[
\Delta = \min(\Delta, C - p_j - r_j)
\]
while \( \Delta > 0 \) and \( \beta \geq \alpha \) do
\[
\text{if } \Delta \geq p_h \text{ then}
\]
\[
\alpha = \alpha + \alpha_h; \quad \beta = \beta - \beta_h
\]
\[
S = S - p_h; \quad C = C - p_h
\]
\[
\Delta = \Delta - p_h
\]
\[
h = h + 1
\]
\[
\text{else}
\]
\[
S = S - \Delta; \quad C = C - \Delta
\]
\[
\Delta = 0
\]
\[
\text{endif}
\]
endwhile
\[
j = j + 1
\]
endwhile

Jobs 1,…, \( j-1 \) are scheduled between \( S \) and \( C \) without idle time. Jobs \( j,\ldots, n \) are scheduled at earliest after \( C \).

Time complexity of Algorithm 1 is \( O(n) \). Even, if there are two loops, the instructions of the inner loops are executed at most \( n \) times during the whole algorithm since \( h \) is increased each time (or \( \Delta \) is set to 0, which means that the algorithm stops).

3. Computational Results

Codes were written in C language and tests were conducted on a Intel Xeon with a 3.0 GHz processor and 8 Gb RAM. Initially the computational tests using the proposed heuristics were conducted using 50 benchmark problems proposed by Sarper (1995), and whose results were better than those obtained by this author. In these problems the common due dates are generated in tight and loose intervals, calculated using the sum of processing times of the jobs, without separating restrictive from unrestrictive due dates. However, if unrestrictive due dates are generated, the problem can be optimally solved in polynomial time using the algorithm of Kanet (1981) (see Section 1 for the description of this method). This affirmation and other results on the complexity of multi-stage scheduling problems with earliness and tardiness penalties can be found in Lauff and Werner (2004).

Due to the dimension of the problems proposed by Sarper (\( n \leq 20 \)) and aiming to use a benchmark with restrictive due dates, test problems were generated according to Sakuraba et al. (2006). These authors applied Kanet’s algorithm to obtain an upper bound of the restrictive...
The bounds were used to generate problems with due dates equal to these values multiplied by \( h = 0.2, 0.4, 0.6 \) and 0.8. This factor indicates how jammed the production line is at the beginning of the schedule, i.e., \( h \) defines how restrictive is the common due date. For each of the five number of jobs \( n = 20, 50, 100, 200, \) and 500) ten different instances were generated under four different scenarios of due dates, totaling 200 problems. Processing time was uniformly distributed between 1 and 99.

The proposed heuristics were compared with heuristic H3 presented by Sarper (1995), which was the one with best results according to the author. This method has two phases; the sequence generation (see Section 1) and the definition of the final schedule through a model presented by Sarper. To realize this comparison, the sequence phase of H3 was re-implemented and CPLEX 10.0 was used to solve the model. The results of the tests are shown below, after the description of the sequencing heuristics evaluated.

**Scheduling Heuristic (SH)**

In the proposed heuristic, the job insertion procedure of NEH heuristic (Nawaz et al., 1983) was used: jobs are ordered according to a rule and then inserted in all possible positions of the previous sequence, keeping the best sequence for the next iteration. This procedure will therefore be called PNEH, and Algorithm 1 is applied to schedule and compare the partial sequences. Differently from the heuristic H3 proposed by Sarper, SH uses an optimal algorithm to generate these schedules. Moreover, several sequencing rules, described on Table 1, are evaluated to construct the initial priority list.

<table>
<thead>
<tr>
<th>Rule n°</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LPT using processing times on the first machine.</td>
</tr>
<tr>
<td>2</td>
<td>LPT using processing times on the second machine.</td>
</tr>
<tr>
<td>3</td>
<td>LPT using the sum of processing times on both machines.</td>
</tr>
<tr>
<td>4</td>
<td>SPT (Shortest Processing Time) using processing times on the first machine.</td>
</tr>
<tr>
<td>5</td>
<td>SPT using processing times on the second machine.</td>
</tr>
<tr>
<td>6</td>
<td>SPT using the sum of processing times on both machines.</td>
</tr>
<tr>
<td>7</td>
<td>Non-increasing order of ([p_{i2} - p_{i1}]).</td>
</tr>
<tr>
<td>8</td>
<td>Using Johnson’s rule (Johnson, 1954).</td>
</tr>
<tr>
<td>9</td>
<td>Using Kanet’s sequencing rule (Kanet, 1981) with processing times on the second machine.</td>
</tr>
</tbody>
</table>

The complete process of initial sequencing of the jobs and the subsequent use of PNEH with Algorithm 1 to evaluate the sequences (as presented in Figure 1) will be called SH\(_\#\) (where \# is the number of the rule used).
Initial sequencing of the jobs according to one of the rules
Use of PNEH with Algorithm 1 to evaluate sequences
Final solution

Figure 1 - SH working structure.

In order to measure the performance of the heuristics proposed, the RDI (Relative Deviation Index) was used. The RDI is calculated as follows:

$$RDI = \frac{CV - BV}{WV - BV},$$

where $CV$ is the compared value, i.e., the solution of the heuristic being evaluated and $BV$ and $WV$ represent, respectively, the best and worst values among the solutions found by all the heuristics evaluated for that problem. The best results correspond to the smallest values of RDI.

Table 2 shows the results of the tests through mean values of RDI and the number of best solutions for each dimension. Each cell of the table represents the average RDI for the 40 problems of the corresponding size.

<table>
<thead>
<tr>
<th>N° of Jobs</th>
<th>Sarper</th>
<th>SH₁</th>
<th>SH₂</th>
<th>SH₃</th>
<th>SH₄</th>
<th>SH₅</th>
<th>SH₆</th>
<th>SH₇</th>
<th>SH₈</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.67</td>
<td>0.63</td>
<td>0.65</td>
<td>0.60</td>
<td>0.07</td>
<td>0.23</td>
<td>0.06</td>
<td>0.40</td>
<td>0.25</td>
</tr>
<tr>
<td>50</td>
<td>0.74</td>
<td>0.58</td>
<td>0.71</td>
<td>0.71</td>
<td>0.12</td>
<td>0.09</td>
<td>0.05</td>
<td>0.39</td>
<td>0.54</td>
</tr>
<tr>
<td>100</td>
<td>0.87</td>
<td>0.48</td>
<td>0.84</td>
<td>0.73</td>
<td>0.11</td>
<td>0.07</td>
<td>0.05</td>
<td>0.43</td>
<td>0.62</td>
</tr>
<tr>
<td>200</td>
<td>0.79</td>
<td>0.39</td>
<td>0.76</td>
<td>0.57</td>
<td>0.12</td>
<td>0.05</td>
<td>0.05</td>
<td>0.38</td>
<td>0.53</td>
</tr>
<tr>
<td>500</td>
<td>0.77</td>
<td>0.40</td>
<td>0.75</td>
<td>0.57</td>
<td>0.12</td>
<td>0.05</td>
<td>0.05</td>
<td>0.37</td>
<td>0.53</td>
</tr>
<tr>
<td>Mean</td>
<td>0.77</td>
<td>0.50</td>
<td>0.74</td>
<td>0.64</td>
<td>0.11</td>
<td>0.10</td>
<td>0.05</td>
<td>0.40</td>
<td>0.54</td>
</tr>
</tbody>
</table>

N° of Best Solutions: 0 1 0 0 32 62 83 0 0 24

Analyzing the results presented in Table 2, it can be noticed that SH₆ provides the best results. SH₃ and SH₄ have the second and third best results, respectively. These results show the importance of prioritizing jobs with shorter processing times for the PNEH, since the best performance heuristics use the SPT criterion for their initial sequencing. It can also be noticed that all SH₄ heuristics outperform Sarper’s H3 in terms of mean RDI.

The performance of SH₆ will also be compared with Sarper’s results through the percentage difference, which can be calculated by the expression:

$$\text{Percentage difference} = \frac{PH - CH}{CH} \times 100,$$

where $PH$ represents the result of the proposed heuristic and $CH$ represents the result of the compared heuristic H3. Table 3 shows these results.
Table 3 – Comparison between the results of SH₆ and Sarper (1995).

<table>
<thead>
<tr>
<th>Percentage Difference</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>20</td>
<td>-5.22</td>
</tr>
<tr>
<td>50</td>
<td>-12.45</td>
</tr>
<tr>
<td>100</td>
<td>-15.98</td>
</tr>
<tr>
<td>200</td>
<td>-17.01</td>
</tr>
<tr>
<td>500</td>
<td>-17.86</td>
</tr>
</tbody>
</table>

It can be observed that an average gain of 12.54% was obtained. In general, this gain increases with the number of jobs and is larger for more restrictive due dates. Table 4 shows the mean CPU time for a problem using SH₆ and Sarper’s H₃ heuristic for problems with 100 or more jobs. CPU times using SH₆ for problems with n < 100 were shorter than or equal to 0.001 second and were not considered here.

Table 4 - Comparison between CPU times of the heuristics (seconds).

<table>
<thead>
<tr>
<th>Number of Jobs</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>H₃</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0.166</td>
<td>0.176</td>
<td>0.23</td>
<td>0.49</td>
</tr>
<tr>
<td>200</td>
<td>0.99</td>
<td>1.11</td>
<td>1.81</td>
<td>5.83</td>
</tr>
<tr>
<td>500</td>
<td>8.98</td>
<td>13.34</td>
<td>41.18</td>
<td>190.54</td>
</tr>
<tr>
<td>SH₆</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0.005</td>
<td>0.006</td>
<td>0.006</td>
<td>0.007</td>
</tr>
<tr>
<td>200</td>
<td>0.040</td>
<td>0.045</td>
<td>0.050</td>
<td>0.054</td>
</tr>
<tr>
<td>500</td>
<td>0.58</td>
<td>0.69</td>
<td>0.76</td>
<td>0.83</td>
</tr>
</tbody>
</table>

SH₆ has a CPU time significantly shorter than H₃ in all cases. This fact occurs because of the linear programming model that is applied to the end of the heuristic H₃. Excluding the model, the CPU times gets closer to the ones of SH, but the results get worse.

4. Conclusions

In this paper, we considered the job scheduling problem of minimizing the mean of absolute deviation from a common due date in a two-machine flowshop scenario. Scheduling heuristics were developed considering properties of the problem to improve the results found in the literature.

Initially, a job scheduling algorithm (Algorithm 1) that obtains an optimal schedule for a given job sequence was presented. Algorithm 1 was used with a job insertion procedure - PNEH - to generate a group of heuristics that differ on the initial job sequencing rule. The results of those heuristics were compared with the heuristic of best performance proposed by Sarper (1995), using problems with up to 500 jobs. The best performance was obtained with the PNEH associated with SPT rule (heuristic SH₆), achieving a mean gain of 12.54% over Sarper’s heuristic. These good results are expected since the proposed methods apply an optimal algorithm to schedule all the considered sequences.

As future works, the performance of the heuristic described here may be evaluated in more general cases on flowshops with m machines.
Acknowledgments

This research has been financially supported by FAPESP - “Fundação de Amparo à Pesquisa do Estado de São Paulo” and by CNPq - “Conselho Nacional de Desenvolvimento Científico e Tecnológico”.

References