A case study in a two-stage hybrid flow shop with setup time and dedicated machines

Hung-Tso Lin\textsuperscript{a}, Ching-Jong Liao\textsuperscript{b,*}

\textsuperscript{a}Department of Industrial Engineering and Management, National Chin-Yi Institute of Technology, Chungli, Taiwan
\textsuperscript{b}Department of Industrial Management, National Taiwan University of Science and Technology, 43 Keelung Road Section 4, Taipei 106, Taiwan

Received 21 August 2001; accepted 2 January 2003

Abstract

In this paper we address a scheduling problem taken from a label sticker manufacturing company. The production system is a two-stage hybrid flow shop with the characteristics of sequence-dependent setup time at stage 1, dedicated machines at stage 2, and two due dates. The objective is to schedule one day’s mix of label stickers through the shop such that the weighted maximal tardiness is minimized. A heuristic is proposed to find the near-optimal schedule for the problem. The performance of the heuristic is evaluated by comparing its solution with both the optimal solution for small-sized problems and the solution obtained by the scheduling method currently used in the shop. As the heuristic is beneficial to the company, it will be implemented in the near future.

Keywords: Two-stage hybrid flow shop; Sequence-dependent setup time; Dedicated machine; Weighted maximal tardiness

1. Introduction

The hybrid flow shop, also known as the flow shop with multiple processors (FSMP), has been extensively studied in the literature. A recent overview on FSMP research is given by Linn and Zhang (1999). However, most of the considered problems are theoretical models. In this paper, we address a practical FSMP problem taken from a label sticker manufacturing company.

The case we are dealing with presents similarities with some of the two-stage FSMP. With respective to the two-stage FSMP, Narasimhan and Panwalkar (1984) considered a real-life FSMP with one machine at stage 1 and two machines at stage 2. The CMD (cumulative minimum deviation) rule was suggested for reducing the sum of machine idle time and in-process job waiting time. Later, Narasimhan and Mangiameli (1987) proposed the GCMD (generalized cumulative minimum deviation) rule, which is an extension of the CMD rule, for the FSMP with five criteria. In their system, the material is processed continuously at stage 1, consisting of multiple and identical machines, and then batch processed on the multiple repetitive machines at stage 2. In addition to the CMD criterion, the makespan ($C_{\text{max}}$) was frequently used as the criterion for the FSMP in the literature.

\*Corresponding author. Tel.: 886-2-27376337; fax: 886-2-27376344.
E-mail address: cjl@im.ntust.edu.tw (C.-J. Liao).
Gupta (1988a) showed that the two-stage FSMP problem is NP-hard, and developed a heuristic in finding a minimum makespan schedule of a special case when there is only one machine at stage 2. Sriskandarajah and Sethi (1989) presented four algorithms for the FSMP with the makespan performance. The algorithms were analyzed in the worst and average case performance contexts. Gupta and Tunc (1991) proposed two heuristics to find a minimum makespan schedule for the case when there is only one machine at stage 1. The global lower bounds on the makespan were also discussed. Deal and Hunsucker (1991) studied the FSMP with identical number of machines at the two stages. A lower bound calculation for the makespan was introduced and employed to evaluate the performance of three job sequencing rules in conjunction with a FIFO (first-in, first-out) manner.

Research work has also been done in the generalized FSMP case where the number of stages and the number of machines at each stage are not restricted. The logic on how the rules in these papers were developed will be of great value to us in developing good heuristic. Kochhar and Morris (1987) presented heuristics to minimize the mean flowtime for the scheduling problem, which consists of two sub-problems: entry point sequencing and dispatching. Various optimization techniques, including myopic and local search methods, and dispatching methods, trying to minimize the effects of setup time and blocking, were investigated for the two sub-problems, respectively. Wittrock (1985, 1988) proposed an algorithm primarily to minimize the makespan and secondarily to minimize the queue. The basic approach of the algorithm is to decompose the problem into three sub-problems, known as machine allocation, sequencing, and timing, each of which is solved by a heuristic. The following generalized FSMP studies all used the makespan as the criterion. Ding and Kittichartphayak (1994) developed three heuristics, in which a job sequence is first determined for the first stage, and then the remaining stages follow the same sequence. A heuristic, known as FLOWMULT, was developed by Santos et al. (1995a). The heuristic uses each of the n! different permutations of jobs as a starting list at the first stage, and employs a FIFO manner as the sequencing procedure between the subsequent stages. A significant drawback of FLOWMULT lies in its computation time since its order of computation is at least factorial in nature. However, FLOWMULT shows that the best of the n! sequences, with a FIFO manner, is often optimal (or near-optimal). This significantly reduces the search space for the FSMP environment with a makespan objective. Also, Santos et al. (1996) evaluated four currently existing methods used in the pure flow shop, known as Palmer, Campbell et al., Gupta, and Dannenbring heuristics, for the FSMP environment. To assess the quality of heuristic solutions, Brah and Hunsucker (1991) developed a branch-and-bound algorithm for an FSMP, and Santos et al. (1995b) presented global lower bounds on the FSMP problem.

Next, we briefly review the related research on parallel machine (PM) scheduling problems, which is an important characteristic of our problem. Li and Cheng (1993) studied the PM scheduling problem with an objective of minimizing the maximum weighted absolute lateness. They showed that the min–max scheduling problem is NP-hard and proposed two greedy heuristics to solve the problem. Later, Cheng et al. (1995) discussed the application of genetic algorithms to the same problem. Cheng and Kovalyov (1999) studied the PM scheduling problem with an objective of minimizing the maximum absolute lateness. For the special case of PM scheduling problem when the due dates must increase in the same sequences as the job release times, called agreeable due dates, Li (1995) presented a heuristic algorithm to minimize the number of late jobs. Complete reviews of PM scheduling problems can be found in Lawler et al. (1993) and Cheng and Sin (1990).

2. The production system

2.1. The manufacturing process

The production system is a two-stage hybrid flow shop. Stage 1 consists of a single high speed machine (called calender) which is used to glue the
surface material and liner together to produce the label stickers. Stage 2 has two types of cutting machines, each consisting of two identical machines. One type of the machines slits the label stickers into specified width and winds on the rolls, and the other type cuts the label stickers into sheets, which are in conformity with the required size (i.e., unit length and width). We refer to the two types of cutting machines as CM1 and CM2, respectively. Depending on the requirement of customer orders, each job is processed on either a CM1 or a CM2 machine at stage 2. The work-flow of the production system is depicted in Fig. 1.

When the calender is changed over from jobs in one class to jobs in a new class, a setup time, which depends on both the previous and the current classes of jobs, is required for the changeover task. At stage 2, the setup time is sequence-independent and relatively minor, and hence it is included in the processing time. In addition, the transfer time is negligible because the distance between the machines at the two stages is short.

2.2. The classes and the setup time

The primary factor in classifying the jobs is the adhesive base. Currently, there are six different adhesive bases (g1, g2, ..., g6) used in the shop, and hence the jobs are grouped into six classes (C1, C2, ..., C6). One of the six adhesive bases is selected to match up to the cohesion, adhesion, and release forces required by the customers. According to adhesive specifications, the temperature is set at 120, 130, 110, 130, 120, and 130 (in °C) for the six adhesive bases, respectively.

The setup tasks include changing the adhesive base and adjusting the operation temperature. For the task of changing adhesive base, if the ingredient of the preceding adhesive base is incompatible with the successor, the predecessor is cleaned away from the adhesive container and then the successor is flowed in; if they are compatible, the cleaning procedure is omitted. From the standpoint of chemistry, g1, g3, and g5 are compatible with g2, g4, and g6, respectively, and hence g1 and g2 can be grouped into G1, g3 and g4 into G2, and g5 and g6 into G3. Table 1

![Fig. 1. Work-flow of the production system.](image-url)
lists the classification scheme of the label stickers. Most of the time for changing the adhesive base is spent on cleaning down the adhesive container. The times to clean away the adhesive bases in G1, G2, and G3 are predetermined as 6, 8, and 10 (in min), respectively. In adjusting the operation G2, and G3 are predetermined as 6, 8, and 10 min. To complete a cut cycle is about 0.05 min.

Based on the above data, the produced length, $W_j$, for $J_j$ belonging to CM1 can be calculated as $Q_j = 1.03(q_j/((W_j - 1)/w_j))$, where $W_j$ (≤ 47 in) is the actual produced width and 1 (in) is the allowed margin for cutting. $W_j$ is determined with the consideration of $w_j$ so that $(W_j - 1)/w_j$ is an integer. The processing time of $J_j$ at stages 1 and 2, denoted by $t_{j1}$ and $t_{j2}$, respectively, can be determined by $t_{j1} = Q_j/v$ and $t_{j2} = 2 + Q_j/60 + 1\cdot\lfloor Q_j/200\rfloor$. For $J_j$ belonging to CM2, the required total length can be derived by $q_j = h_j/l_j$. $Q_j$ and $t_{j1}$ are calculated as above. The processing time at stage 2 can be calculated as $t_{j2} = 2 + 0.05\lfloor h_j/((W_j - 1)/w_j)\rfloor$.

### 2.4. The scheduling objective

The company segmented the market into three regions, denoted by $R_1$, $R_2$, and $R_3$, and established three, four, and three offices, respectively, to do business with the customers. The arrival jobs in a day, ordered from the 10 offices, are released to the shop in the next day. The problem to be addressed is to schedule one day’s mix of label stickers through the shop. The shop works two shifts, from 8 a.m. to 4 p.m. and from 4 p.m. to midnight, each day. For the jobs ordered from $R_1$, $R_2$, and $R_3$, the delivery times are set at 4 a.m., 7 a.m. (in the succeeding day), and 6 p.m. (in the currently working day), respectively. With the consideration of delivery and working times, the jobs ordered from $R_3$ and those from $R_1$ and $R_2$ should be completed before 6 p.m. and midnight, respectively. Since 6 p.m. is the delivery time of the trucking company and midnight is the off-duty time of the shop, we refer to 6 p.m. as the exogenous due-date, $d_1$, and midnight as the internal due-date, $d_2$. Waiting time of the deliverers and overtime of the operators at stages 1 and/or 2 will be resulted if $d_1$ and $d_2$ are violated, respectively. For convenience, let $V_1$ and $V_2$ be the

### Table 2

<table>
<thead>
<tr>
<th>From</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0</td>
<td>10</td>
<td>26</td>
<td>16</td>
<td>6</td>
<td>16</td>
</tr>
<tr>
<td>C2</td>
<td>20</td>
<td>0</td>
<td>46</td>
<td>6</td>
<td>26</td>
<td>6</td>
</tr>
<tr>
<td>C3</td>
<td>18</td>
<td>28</td>
<td>0</td>
<td>20</td>
<td>18</td>
<td>28</td>
</tr>
<tr>
<td>C4</td>
<td>28</td>
<td>8</td>
<td>40</td>
<td>0</td>
<td>28</td>
<td>8</td>
</tr>
<tr>
<td>C5</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>20</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>C6</td>
<td>30</td>
<td>10</td>
<td>50</td>
<td>10</td>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>
sets including the jobs ordered from $R_3$ (i.e., with $d_1$) and those from $R_1$ and $R_2$ (i.e., with $d_2$). Denote by $n_1$ and $n_2$ the numbers of jobs in $V_1$ and $V_2$, respectively. Let $V = V_1 \cup V_2$ and $n = n_1 + n_2$. The maximal exogenous tardiness, internal tardiness at stages 1 and 2, denoted by ET, IT$_1$, and IT$_2$, respectively, are calculated as

$$\text{ET} = \max_{J_i \in V_1} \{0, C_{j_2} - d_1\},$$

$$\text{IT}_1 = \max_{J_i \in V} \{0, C_{j_1} - d_2\},$$

and

$$\text{IT}_2 = \max_{J_i \in V} \{0, C_{j_2} - d_2\},$$

where $C_{j_g}$ is the completion time of $J_j$ at stage $g$. The objective is to complete the jobs such that the sum of waiting cost and overtime cost is minimized. Denote the unit waiting cost and the unit overtime cost at stages 1 and 2 by $\alpha$, $\beta_1$, and $\beta_2$ (in dollars/min), respectively. Then, the performance measure weighted maximal tardiness (WT$_{\text{max}}$) can be expressed as

$$\text{WT}_{\text{max}} = \alpha \times \text{ET} + \beta_1 \times \text{IT}_1 + \beta_2 \times \text{IT}_2.$$

We now explain the value settings of $\alpha$, $\beta_1$, and $\beta_2$. There are three deliverers to perform the delivery task. The average salary per month for a deliverer is $40,000$ (in New Taiwan Dollars) and it is charged approximately as much for the waiting time by the trucking company. Thus, the value of $\alpha$ can be calculated as

$$\alpha = (3 \ \text{deliverers} \times 40,000 \ \text{dollars/month}$$

$$+ 25 \ \text{days/month} \times 480 \ \text{min/day}) \times 2 = 20.0.$$

At stages 1 and 2, there are three and four workers, respectively, to perform the production tasks. The average salaries per month for a worker at stages 1 and 2 are $35,000$ and $30,000$, respectively, and it is charged 1.33 times as much for the overtime. Thus, the values of $\beta_1$ and $\beta_2$ can be calculated as

$$\beta_1 = (3 \ \text{workers} \times 35,000 \ \text{dollars/month}$$

$$+ 25 \ \text{days/month} \times 480 \ \text{min/day}) \times 1.33 = 11.6$$

and

$$\beta_2 = (4 \ \text{workers} \times 30,000 \ \text{dollars/month}$$

$$+ 25 \ \text{days/month} \times 480 \ \text{min/day}) \times 1.33 = 13.3.$$

For convenience in analysis, the three values are normalized by the sum, yielding the relative weights of $\alpha = 0.446$, $\beta_1 = 0.258$, and $\beta_2 = 0.296$.

2.5. The current scheduling method

The philosophy of the current method used in the shop is to schedule jobs with $d_1$ before jobs with $d_2$ and to complete jobs at stage 1 as early as possible so that they can be transferred to stage 2. The production sequence of the $n$ jobs at stage 1, $(J_{[1]} - J_{[2]} - \cdots - J_{[n]})$, is determined by the analogue of the heuristic developed by Gupta (1988b), called the Gupta-type approach, which is described as follows:

Step 1: Select $J_{[1]} = \min_{J_i \in V_1} \{t_{j_1}\}$. Delete $J_{[1]}$ from $V_1$. Let $p = 1$.

Step 2: Let $J_p$ be the job in position $p$. Set $p = p + 1$. Select $J_{[p]} = \min_{J_i \in V_2} \{s_{j_2} + t_{j_1}\}$. Delete $J_{[p]}$ from $V_1$.

Step 3: Perform Step 2 recursively until $V_1 = \emptyset$.

Step 4: Let $J_2$ be the job in position $p$. Set $p = p + 1$. Select $J_{[p]} = \min_{J_i \in V_2} \{s_{j_2} + t_{j_1}\}$. Delete $J_{[p]}$ from $V_2$.

Step 5: Perform Step 4 recursively until $V_2 = \emptyset$.

When a queue of jobs for the next available machine at stage 2 exists, the jobs are processed in a FIFO manner.

3. The addressed scheduling problem and proposed scheduling rules

By considering only CM$_1$ at stage 2 and setting the values of $\alpha$, $\beta_1$, and $d_2$ equal to zero and the value of $\beta_2$ equals to 1, the problem becomes a special case of our problem. The performance measure

$$\text{WT}_{\text{max}} = \max_{J_i \in V} \{C_{j_2}\} = C_{\text{max}}.$$

As Gupta (1988a) has shown that this special case of the two-stage FSMP scheduling problem is NP-hard, our problem is also NP-hard.

Due to the complexity of our problem, we dedicate our efforts to develop a heuristic to find an approximate solution. The proposed scheduling rules for the problem consist of the following elements:

1. Determine the production sequences at stage 1.
(2) Dispatch the jobs in the queue at stage 2.
(3) Develop and improve the schedules.

3.1. Determine the production sequences at stage 1

We first arrange the jobs with \( d_1 \) before the jobs with \( d_2 \) for the consideration of due dates. The following three sequencing methods are then employed to produce three initial sequences:

(i) Arrange the jobs to minimize the total setup time (TST).
(ii) Use the concept of Approximate Algorithm 1 proposed by Gupta and Darrow (1986) (modified GD).
(iii) Process the job with the longest processing time at stage 2 first (LPT).

The TST method is proposed to minimize the makespan at stage 1. According to the TST method, the jobs in a given class are processed together in arrival order and the class sequences which follow C3–C1–C5–C6–C2–C4 and C4–C2–C6–C5–C1–C3 for the jobs with \( d_1 \) and \( d_2 \), respectively, are the best sequences, which can be obtained by full enumeration. The concept of Approximate Algorithm 1 (Gupta and Darrow, 1986) is employed as it is proposed to reduce the makespan at stage 2 for the two-machine flowshop problem, the setup time is sequence-independent with sequence-dependent setup time. In our problem, the setup time at stage 2 is first (LPT).

FIFO is the discipline usually utilized in the FSMP environment (Santos et al., 1996). LPT and SPT are the alternatives suggested by Santos et al. (1995a).

3.2. Dispatch the jobs at stage 2

The following three dispatching methods are employed:

(i) Use the FIFO manner.
(ii) Process the job with the longest processing time at stage 2 first (LPT).
(iii) Process the job with the shortest processing time at stage 2 first (SPT).

For a sequencing method, an initial sequence, \( S_0 \), is first generated. Then \( r \) adjusted sequences, \( S_k \) \( (k = 1, 2, \ldots, r) \), are generated from \( S_{k-1} \) by employing the pairwise comparison procedure. In the pairwise comparison procedure a sequence with smaller value of makespan will be used to update the current sequence, and hence the value of maximal tardiness will also be smaller.

\[ S_{k-1}' = \{ (J_{[1]}', \ldots, J_{[n]}') \} \text{ from } S_k \text{ by moving the job in position } p \text{ forward to position } q \text{.} \]

3.3. Develop and improve the schedules

\[ r + 1 \] sequences, \( S_k \) \( (k = 0, 1, \ldots, r) \), are generated for a sequencing method. Each of the \( r + 1 \) sequences is used to develop a schedule. If the schedule results in tardiness, it will be improved by altering the production sequence. We incorporate the concept of tabu search method into the following procedure, called the recursive alteration procedure, to develop and improve the schedule:

**Step 1:** Denote the sequence of \( S_k \) by \( (J_{[1]} - J_{[2]} - \cdots - J_{[n]} - J_{[n+1]} - \cdots - J_{[|S|]}) \). Let the tabu list \( L = \emptyset \).

**Step 2:** Process the jobs at stage 1 by the order arranged in \( S_k \) and dispatch the jobs at stage 2 by the specified dispatching method. Calculate its WT\(_{\text{max}}\) value as

\[ \text{WT}_{\text{max}}(S_k) = 0.446 \text{ ET} + 0.258 \text{ IT}_1 + 0.296 \text{ IT}_2. \]

**Step 3:** If \( \text{WT}_{\text{max}}(S_k) = 0 \), an optimal schedule is obtained and the procedure is terminated; otherwise, continue.

**Step 4:** Alter \( S_k \) by the following:

(a) Let \( p = 2, q = 1 \).
(b) Generate the altered sequence \( S_k' = (J_{[1]}' - J_{[2]}' - \cdots - J_{[n]}') \) from \( S_k \) by moving the job in position \( p \) forward to position \( q \). That is, let \( J_{[q]}' = J_{[p]}, J_{[p+1]}' = J_{[q]} \) \( (q \leq l \leq p - 1) \), and \( J_{[q]}' = J_{[i]} \) \( (1 \leq i \leq q - 1 \) and \( p + 1 \leq i \leq n) \).
(c) If \( S_k' \in L \), enter Step 4(d); otherwise, perform the following steps:
   (i) Develop the schedule by using \( S_k' \) and the specified dispatching method.
(ii) Calculate the WTmax value of the schedule.

(iii) Add $S'_k$ to $L$ in the following way: If the number of sequences included in $L$, denoted by $y$, is less than seven, $S'_k$ is put on the position $y + 1$; otherwise, $L$ is shifted to the left and $S'_k$ is put on the position seven.

(d) Let $q = q + 1$.

(e) Perform Steps 4(b) to (d) recursively until $q = p - 1$.

(f) Let WTmax($S_k$) be the minimum of the WTmax values obtained thus far and $S_k$ be the associated schedule.

(g) Let $p = p + 1$ and reset $q = 1$.

(h) Perform Steps 4(b) to (g) recursively until $p = n$.

**Step 5:** Use the concept of tabu-type approach, perform Steps 3 and 4 recursively up to seven times or until WTmax($S_k$) = 0. The resulted schedule is the schedule generated by the method.

We now elaborate the steps in detail. In Steps 1 and 2, the sequence $S_k$ is used to develop the schedule. If the schedule results in tardiness, one of the three cases occurs:

1. ET > 0 and IT1 + IT2 = 0,
2. ET > 0 and IT1 + IT2 > 0, and
3. ET = 0 and IT1 + IT2 > 0.

Because we have initially processed the jobs with $d_1$ before those with $d_2$ at stage 1, altering the sequence of jobs with $d_1$ (i.e., the jobs in the first $n_1$ positions) may reduce the WTmax value in case (1). In cases (2) and (3), altering the sequence of jobs with $d_1$ and/or with $d_2$ (i.e., the jobs in the last $n_2$ positions), and/or processing the jobs with $d_2$ before those with $d_1$ may reduce the WTmax value.

To generalize the alteration, it is performed by moving $J_{pi}(2 \leq p \leq n)$, to all the positions before it. The best WTmax value along with the associated schedule obtained thus far is saved as WTmax($S_k$) and $S_k$. The alteration is performed until $p = n$ in Step 4. Our improvement procedure is based on the concept of tabu search method. A function, called a move, which transforms a solution into another solution is performed in Step 4. For any solution $S_k$, the moves generate a set of solutions NH($S_k$), called the neighborhood of $S_k$. Starting from an initial solution, this search method iteratively moves from the current solution to the best solution in NH($S_k$) until a stopping criterion specified in step 5 is satisfied. Glover (1977) suggested that an accepted solution should be considered as tabu for seven moves, after which the tabu status is removed and the solution is again allowable. Therefore, we set the length of tabu list as seven in Step 4(c)(iii) and, for computational reasons, perform the procedure up to seven times or until WTmax($S_k$) = 0 in Step 5.

4. The experiments

4.1. The job data

The job data from the past six months in the shop were collected to be used as the empirical distributions in the experiments. There were 146 working days (i.e., 146 samples) in the period. The data can be summarized as follows:

(i) The percentages that a job belongs to C1, C2, C3, C4, C5, and C6 are 0.20, 0.12, 0.17, 0.26, 0.14, and 0.11, respectively.

(ii) The percentages that a job belongs to CM1 and CM2 are 0.65 and 0.35, respectively.

(iii) The percentages that a job is included in $V_1$ (with $d_1$) and $V_2$ (with $d_2$) are 0.45 and 0.55, respectively.

(iv) For a job belonging to CM1, the minimal and maximal length are 2000 and 5000 (in yards), and the width ranges from 9 to 30 (in in). For a job belonging to CM2, the minimal and maximal numbers of sheets are 4000 and 7500, and both of the unit length and width range from 9 to 18 (in in).

(v) The minimal and maximal numbers of jobs produced in a working day are 24 and 40.

In the experiments, the uniform distribution is used for (iv) and (v).
4.2. The pilot experiments

Combining the three sequencing methods at stage 1 and the three dispatching methods at stage 2 results in nine scheduling rules. We conducted the pilot experiments to test the performance of the rules. Clearly, the more the number of sequences used to develop and improve the schedules, the better the solution and, in nature, the more the computation time. To explore the performance of the scheduling rules under different number of sequences, we set \( r \) values as 1, \( \lfloor (x/10)n \rfloor \) (\( x = 1, 2, \ldots, 9 \)), and \( n \), respectively. For each of the rules, five problems were randomly drawn from the empirical distributions described in Section 4.1. The setup time used the industrial data listed in Table 2. For each of the test problems, the best WT\(_{\text{max}}\) value obtained by using each of the \( r + 1 \) sequences, \( S_k \) (\( k = 0, 1, \ldots, r \)), to develop and improve the schedule recursively served as the approximate solution. The results show that, generally speaking, the WT\(_{\text{max}}\) value declines with increasing of \( r \) and reaches the flattening section at \( r = \lfloor 0.3n \rfloor \) from a judgement in rough. Hence, we set \( r = \lfloor 0.3n \rfloor \) in the following experiments. Fig. 2 illustrates, for instance, the performance of the rule composed of TST and FIFO under different \( r \) values.

For each of the rules, another 50 problems were randomly drawn to compare their performance. The average of the 50 approximate solutions for each of the scheduling rules is summarized in Table 3, which shows that the rule combining TST and FIFO performed best. It is, therefore, selected as the proposed heuristic. Table 3 also shows that TST is the best sequencing method at stage 1, which implies that reducing the makespan at stage 1 initially is contributive to the objective.

4.3. Evaluate the heuristic

To test the effectiveness of the heuristic, experiments for small-sized problems (\( n = 7, \ldots, 10 \)) were conducted. Denote by \( P_1 \) and \( P \) the time at which the jobs in \( V_1 \) and \( V \) are completed. The completion time can be approximately calculated as

\[
P_1 = (c_1 - 1)\bar{s} + \sum_{j \in V_1} t_{j1} + \min_{j \in V_1} \{ t_{j2} \}
\]

and

\[
P = (c - 1)\bar{s} + \sum_{j \in V} t_{j1} + \min_{j \in V} \{ t_{j2} \},
\]

where \( c_1 \) and \( c \) are the numbers of classes of the jobs in \( V_1 \) and \( V \), and \( \bar{s} \) and \( \bar{s} \) are the average setup times of the \( c_1 \) and \( c \) classes, respectively. Then, \( d_1 \) and \( d_2 \) were generated from the uniform
distributions \( P_1(1 - \tau - \frac{1}{2}\rho), P_1(1 - \tau + \frac{1}{2}\rho) \) and \( P(1 - \tau - \frac{1}{2}\rho), P(1 - \tau + \frac{1}{2}\rho) \), respectively, which follows the methods proposed by Potts and Van Wassenhove (1991) and Kim (1995). If \( d_2 \leq d_1 \), \( d_2 \) was re-generated from \( [d_1, P(1 - \tau + \frac{1}{2}\rho)] \). The values for \( \tau \) and \( \rho \) ranged from 0.2 to 0.4 and from 0.4 to 0.8, respectively. Twenty test problems were generated for each combination of \( \tau \) and \( \rho \). The optimal solutions were obtained by a branch-and-bound algorithm (Lin and Liao, 1999). The computational results are summarized in Table 4, where the percentage error is defined as

\[
\frac{\text{heuristic } \text{WT}_{\text{max}} - \text{optimal } \text{WT}_{\text{max}}}{\text{optimal } \text{WT}_{\text{max}}} \times 100\% \]

The average % error of the heuristic is 1.50% with a maximum of 14.75%. The heuristic produced a solution with optimum in approximate 70% (336/480) of the test problems, and within 3% and 5% of optimality in approximate 81% ((336 + 53)/480) and 87% ((336 + 53 + 30)/480), respectively. In general, the results indicate that the heuristic is fairly effective in finding an optimal or a near-optimal solution for small-sized problems.

The same job data in the shop were used in the simulated application to compare the performance of the heuristic solution with the solution obtained by the current scheduling method. The maximal, average, and minimal computation times of the heuristic are 711, 388, and 152 (in s), respectively. The maximal, average, and minimal \( \text{WT}_{\text{max}} \) values resulted by the current method are 336.2, 116.0, and 17.0, respectively. By using these values, the maximum, average, and minimum of the sums of waiting and overtime cost resulted by the current method are calculated as

\[
336.2 \times (20.0 + 11.6 + 13.3) = 15,095.38, \quad 116.0 \times (20.0 + 11.6 + 13.3) = 5,208.40, \quad \text{and} \quad 17.0 \times (20.0 + 11.6 + 13.3) = 763.30, \text{respectively. These are compared to} \quad 11,894.01, 3,125.04, \text{and} \quad 170.62 \text{by the heuristic.}
\]

To explore the effectiveness of the heuristic, we define the percentage improvement as

\[
\frac{\text{current } \text{WT}_{\text{max}} - \text{heuristic } \text{WT}_{\text{max}}}{\text{current } \text{WT}_{\text{max}}} \times 100\% \]

The average % improvement is 47%, with the minimum and maximum of 15% and 77%, respectively. It can be observed that the heuristic performs much better.

5. Conclusions

In this paper we have addressed a real-life scheduling problem in a manufacturing company producing label stickers. The production system is a two-stage hybrid flow shop with the characteristics of sequence-dependent setup time at stage 1, dedicated machines at stage 2, and two due dates. The objective is to schedule one day’s mix of label stickers through the shop such that the weighted maximal tardiness is minimized.

The case treated in this paper does not fall into any category of the two-stage FSMP scheduling problems in the literature, but it bears certain similarities. A heuristic composed of TST at stage 1, FIFO at stage 2, and recursive alteration...
procedure is proposed. As the heuristic is based on the specific requirements of the system, it can effectively improve the performance of the system. The management is currently developing an aggregate production management system, which includes order treatment, scheduling, inventory control, forecasting, and capacity planning modules. As the heuristic is beneficial to the company, it will be used in the scheduling module and implemented in the near future. Although the heuristic is developed for the specific system, it can be used, with appropriate modifications, in other two-stage FSMP scheduling problems with similar features.

### Appendix

For our problem, the Approximate Algorithm 1 of Gupta and Darrow (1986) is modified, called the modified GD method, by letting the setup time at stage 2 equal zero. The method is performed as follows:

**Step 0:** Let \( \sigma \) be an existing initial partial schedule with last job as \( J_d \). Denote the post partial schedule by \( \rho \) and let \( J_e \) be its first job. Let the number of jobs in \( \sigma \) be \( k_1 \) and in \( \rho \) be \( k_2 \). Let \( k = k_1 + k_2 \) and \( \pi \) be the subset of jobs not included in \( \sigma \) and \( \rho \). Initially, set \( \sigma = \rho = \emptyset \), \( k_1 = k_2 = 0 \), and \( \pi = \{ J_1, J_2, ..., J_n \} \). Enter Step 1.

### Table 4
Computational results for the heuristic

<table>
<thead>
<tr>
<th>No. of jobs</th>
<th>( \tau )</th>
<th>( \rho )</th>
<th>Min.</th>
<th>Mean</th>
<th>Max.</th>
<th>( e_0^a )</th>
<th>( e_3^b )</th>
<th>( e_5^c )</th>
<th>Average comp. time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0.2</td>
<td>0.4</td>
<td>0</td>
<td>0.56</td>
<td>4.88</td>
<td>16</td>
<td>2</td>
<td>2</td>
<td>2.20</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>0</td>
<td>1.06</td>
<td>4.93</td>
<td>15</td>
<td>0</td>
<td>5</td>
<td>2.53</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.8</td>
<td>0</td>
<td>0.89</td>
<td>5.20</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>3.22</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.6</td>
<td>0</td>
<td>0.45</td>
<td>3.91</td>
<td>17</td>
<td>1</td>
<td>2</td>
<td>3.13</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.8</td>
<td>0</td>
<td>0.63</td>
<td>6.73</td>
<td>17</td>
<td>1</td>
<td>1</td>
<td>2.90</td>
</tr>
<tr>
<td>8</td>
<td>0.2</td>
<td>0.4</td>
<td>0</td>
<td>1.35</td>
<td>8.64</td>
<td>15</td>
<td>1</td>
<td>0</td>
<td>6.35</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>0</td>
<td>1.13</td>
<td>8.95</td>
<td>16</td>
<td>1</td>
<td>0</td>
<td>5.26</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.8</td>
<td>0</td>
<td>2.33</td>
<td>9.05</td>
<td>13</td>
<td>1</td>
<td>1</td>
<td>5.93</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.6</td>
<td>0</td>
<td>1.48</td>
<td>9.82</td>
<td>14</td>
<td>3</td>
<td>0</td>
<td>6.62</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.8</td>
<td>0</td>
<td>0.93</td>
<td>7.57</td>
<td>14</td>
<td>4</td>
<td>0</td>
<td>5.42</td>
</tr>
<tr>
<td>9</td>
<td>0.2</td>
<td>0.4</td>
<td>0</td>
<td>1.04</td>
<td>11.06</td>
<td>16</td>
<td>1</td>
<td>2</td>
<td>9.03</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>0</td>
<td>2.26</td>
<td>9.73</td>
<td>11</td>
<td>3</td>
<td>2</td>
<td>8.49</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.8</td>
<td>0</td>
<td>2.03</td>
<td>12.30</td>
<td>14</td>
<td>1</td>
<td>1</td>
<td>9.28</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.6</td>
<td>0</td>
<td>2.69</td>
<td>13.69</td>
<td>13</td>
<td>1</td>
<td>1</td>
<td>8.78</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.8</td>
<td>0</td>
<td>1.54</td>
<td>12.08</td>
<td>13</td>
<td>3</td>
<td>3</td>
<td>8.46</td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
<td>0.4</td>
<td>0</td>
<td>2.80</td>
<td>14.75</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>12.89</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>0</td>
<td>2.01</td>
<td>13.46</td>
<td>13</td>
<td>2</td>
<td>2</td>
<td>13.57</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.8</td>
<td>0</td>
<td>2.68</td>
<td>12.93</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>14.03</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.6</td>
<td>0</td>
<td>2.54</td>
<td>10.06</td>
<td>11</td>
<td>3</td>
<td>1</td>
<td>12.17</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.8</td>
<td>0</td>
<td>1.57</td>
<td>11.54</td>
<td>15</td>
<td>2</td>
<td>0</td>
<td>13.85</td>
</tr>
<tr>
<td>Aggregate</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.50</td>
<td>14.75</td>
<td>336</td>
<td>53</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

\( ^a e_0 \) denotes the number of times that the optimal solution is obtained.
\( ^b e_3 \) denotes the number of times that the % error is greater than zero but less than or equal to three.
\( ^c e_5 \) denotes the number of times that the % error is greater than three but less than or equal to five.
Step 1:
(a) Find $J_a$ such that $A = t_{a1} + s_{da} = \min_{j \in \pi} \{t_{j1} + s_{dj}\}$.
(b) Find $J_b$ such that $B = t_{b2} = \min_{j \in \pi} \{t_{j2}\}$.
(c) If $A < B$, let $\sigma = J_a$, $k_1 = k_1 + 1$, and enter Step 2. If $A > B$, let $\rho = J_b\rho$, $k_2 = k_2 + 1$, and enter Step 2. If $A = B$, proceed with Step 1(d).
(d) Case 1. $J_a \neq J_b$. If $\min \{t_{a1} + s_{da}, t_{b2}\} \leq \min \{t_{b1} + s_{db}, t_{a2}\}$, set $\sigma = J_a$, $k_1 = k_1 + 1$; otherwise, set $\rho = J_b\rho$, $k_2 = k_2 + 1$. Enter Step 2.
Case 2. $J_a = J_b$. If $t_{a1} + s_{da} \leq t_{a2}$, set $\sigma = J_a$, $k_1 = k_1 + 1$; otherwise, set $\rho = J_b\rho$, $k_2 = k_2 + 1$. Enter Step 2.

Step 2: If $k = k_1 + k_2 < n - 1$, update $\pi$ and return to Step 1; otherwise, the complete sequence $\sigma\pi\rho$ is obtained.

References


