Research Article

A Heuristic Approach to $n \times m$ Flow Shop Scheduling Problem in Which Processing Times Are Associated with Their Respective Probabilities with No-Idle Constraint

Deepak Gupta and Harminder Singh

Department of Mathematics, Maharishi Markandeshwar University, Mullana, Ambala, Haryana 133-207, India

Correspondence should be addressed to Harminder Singh; harminder.cheema85@gmail.com

Received 20 June 2013; Accepted 30 July 2013

Academic Editors: P. Ekeland and X.-M. Yuan

Copyright © 2013 D. Gupta and H. Singh. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper is an attempt to study general $n \times m$ flow shop scheduling problem in which processing time of jobs is associated with probabilities under no-idle constraint. The objective of this paper is to develop a heuristic algorithm to $n \times m$ flowshop scheduling so that no machine remains idle during working for any given sequence of jobs. The proposed algorithm is simple, and easy to understand and provides an important tool in many practical situations for minimizing the expected hiring cost of the machines for a fixed sequence of job processing. A numerical illustration is also given to justify the proposed algorithm.

1. Introduction

In flow shop scheduling problems, the objective is to obtain a sequence of jobs which when processed on the machines will optimize some well-defined criteria. Every job will go on these machines in a fixed order of machines. The research into flow shop problems has drawn a great attention in the last decades with the aim to increase the effectiveness of industrial production. Johnson [1] gave procedure for finding the optimal schedule for $n$-jobs, two-machine flow-shop problem with minimization of the makespan (i.e., total elapsed time) as the objective. Ignall and Schrage [2] applied Branch and Bound technique for obtaining a sequence which minimizes the total flow time. In addition Adiri and Pohoryles [3] elucidated no-idle scheduling to minimize the sum of completion time. Rajendran and Chaudhuri [4] have given conditions to obtain a sequence which minimizes total flow time subject to minimum makespan in a two-stage flow shop problem. Szwarc [5], Yoshida and Hitomi [6], Anup [7], and so forth, derived the optimal algorithm for two/three or multistage flow shop problems taking into account the various constraints and criteria. Singh et al. [8] associated probabilities with processing time and setup time in their studies. Later, Gupta et al. [9] and Gupta and Singh [10] studied $n \times 2$ general flow shop problem to minimize rental cost under a predefined rental policy in which the probabilities have been associated with processing time on each machine and other scheduling problems by considering various parameters like transportation, idle/waiting operator, and so forth. Narain and Bagga [11–13] studied the flow shop problem with the objective being total rental cost. The total rental cost is minimized when idle time on all the machines is zero.

Under the no-idle situation, machines work continuously without any break; that is, machines should not remain idle once they start processing the first job. The no-idle situation arises in real life world, when machines have to be hired to complete an assignment. Minimization of the total expected hiring cost of the machines would be the objective in these type of situations. The total expected hiring cost of the machines will be at a minimum when idle times on the machines are minimum. Hence, the total expected hiring cost of the machines will be minimum when the idle times of all the machines are zero and under no-idle situation each machine is to be hired for time that is equal to the sum of processing times of all the jobs on it. We are extending the study done by Gupta Deepak including the concept of no-idle scheduling by associating probabilities to the processing time of the jobs. The present paper is an attempt to study the nm general flow shop scheduling with an objective to develop a heuristic algorithm such that no machine remains idle.
2. Practical Situation

Many applied and experimental situations exist in our day-to-day working in factories and industrial production concerns, and so forth, in which different jobs are processed on various machines in a fixed order. For example, in a foundry workshop, the drawing of iron round, cutting, heating, forging, machining, finishing, and packing of finished articles have a fixed order of processing that cannot be altered. Various practical situations also occur in real life when one has got the assignments but does not have one's own machine or does not have enough money or does not want to take risk of investing huge amount of money to purchase machine. Under such circumstances, the machine has to be taken on rent in order to complete the assignments. Hiring of machines is an affordable and quick solution in various production, which is presently constrained by the availability of limited funds due to the recent global economic recession. Hiring enables saving working capital, gives option for having the equipment, and allows upgradation to new technology.

3. Notations and Definitions

The various notations used throughout the paper are as follows:

- \( S \): given fixed sequence of jobs,
- \( M_j \): machine \( j, j = 1, 2, 3, \ldots, m \),
- \( a_{i,j} \): processing time of \( i \)th job on machine \( M_j \),
- \( p_{i,j} \): probability associated to the processing time \( a_{i,j} \),
- \( A_{i,j} \): expected processing time of \( i \)th job on machine \( M_j \),
- \( t_{i,j}(S) \): completion time of \( i \)th job of sequence \( S \) on machine \( M_j \),
- \( l_{i,j}(S) \): idle time of machine \( M_j \) for \( i \)th job in the sequence \( S \),
- \( l_{i,j}(S), l_{j}(S) \): idle time of machine \( M_j \) for \( i \)th job in the sequence \( S \) when machine \( M_j \) starts at latest time \( L_j \).

**Definition 1.** Completion time of \( i \)th job on machine \( M_j \) is denoted by \( t_{i,j} \) and is defined as

\[
t_{i,j} = \max\left(t_{i-1,j}, t_{i,j-1}\right) + a_{i,j} + p_{i,j} \\
= \max\left(t_{i-1,j}, t_{i,j-1}\right) + A_{i,j},
\]

(1)

where \( A_{i,j} \) is expected processing time of \( i \)th job on machine \( j \).

**Definition 2.** Completion time of \( i \)th job on machine \( M_j \) when \( M_j \) starts processing jobs at time \( L_j \) is denoted by \( t_{i,j}^l \) and is defined as

\[
t_{i,j}^l = L_j + \sum_{k=1}^{l} A_{k,j} = \sum_{k=1}^{l} I_{k,j} + \sum_{k=1}^{l} A_{k,j},
\]

(2)

\[
l_{i,j} = \max\left(t_{i,j-1} - t_{i-1,j}, 0\right) \text{ for } j \geq 2.
\]

Also,

\[
t_{i,j}^l = \max\left(t_{i,j}, t_{i,j-1}, t_{i,j-1,1}\right) + A_{i,j}.
\]

**Theorem 3.** The time at which machine \( M_r \) should be taken on rent (or starts processing jobs) to have zero idle time on \( M_r \) is

\[
H_r = \max_{1 \leq k \leq n}\{Y_k\}, \quad r = 2, 3, \ldots, m
\]

where

\[
Y_k = t_{(k-1)}^l - \sum_{i=1}^{k-1} A_{i,r}, \quad k \geq 1,
\]

(5)

\[
Y_1 = t_{1}^l.
\]

**Proof.** Proof is based on mathematical induction. It will be shown that if machine \( M_r \) starts processing jobs at time \( H_r \), then the idle time of \( M_r \) is zero as

For \( r = 2 \),

\[
H_2 = \max_{1 \leq k \leq n}\{Y_k\}
\]

Let \( Y_q = \max_{1 \leq k \leq n}\{Y_k\} \)

Therefore, \( Y_q \geq Y_k \) for \( k = 1, 2, 3, \ldots, n \).

For \( k = 1, Y_q \geq Y_1 \)

Which implies \( H_2 \geq t_{(1,1)} \) or

\[
t_{(1,1)} \leq H_2.
\]

(6)

From (6), if machine \( M_2 \) is taken on rent at time \( H_2 \), then it will start processing the first job without waiting. Therefore, idle time of machine \( M_2 \) for first job is zero when it starts processing jobs at time \( H_2 \) as

\[
Y_q \geq Y_k \text{ for } k = 2, 3, \ldots, n.
\]

\[
Y_q \geq \sum_{i=1}^{k} A_{i,2} \geq Y_2 + \sum_{i=1}^{k} A_{i,2}
\]

that is \( H_2 + \sum_{i=1}^{k-1} A_{i,2} \geq t_{(k-1,2)}^l \) or

\[
t_{(k-1,2)} \geq t_{(1,1)}^l \text{ or } t_{(k,1)} \leq t_{(k-1,2)}^l \text{ for } k = 2, 3, \ldots, n.
\]

(7)

Therefore, \( t_{(k,2)}^l = \max\{t_{(k,1)}^l - t_{(k-1,2),1}^l, 0\} \).

From (7),

\[
t_{(k,2)}^l = 0 \text{ for } k = 2, 3, \ldots, n.
\]

Therefore, the result holds for \( r = 2 \).

Let the result hold for \( r = 5 \).

Now we will also show that the result is also true for \( r = s + 1 \).
Step 1. Calculate the expected processing time $A_{i,j} = a_{i,j} \times p_{i,j}$ for all $i, j = 1, 2, 3, \ldots, m$.

Step 2. For the fixed given sequence $S$, prepare the in-out table for the machine pair $(M_j, M_{j+1})$, $j = 1, 2, 3, \ldots, m - 1$ as two-machine flow shop sequence problem.

Step 3. Compute $K_{j+1}$ for the machine pair $(M_j, M_{j+1})$ by the formula

$$K_{j+1} = t_{n,j+1} - \sum_{i=1}^{n} A_{i,j+1}, \quad j = 1, 2, 3, \ldots, m - 1. \quad \text{(10)}$$

Step 4. Calculate latest time $L_j$ for the machine pair $(M_j, M_{j+1})$ by the formula

$$H_j = H_{j-1} + K_j, \quad j = 3, 4, 5 \ldots m - 1, \quad \text{(11)}$$

$$H_2 = K_2, \quad H_1 = 0.$$  

Step 5. Prepare the in-out table for the machines with latest times $L_j$; the idle time is zero for all machines.

5. Fuzzy Logic Engine

The calculations have been done using C++ program by using different values of processing times and probabilities (Algorithm 1). Based on the results, fuzzy logic engine has been formed in MATLAB fuzzy logic toolbox. The fuzzy logic rule base is formed on the results generated from the C++ program. The membership function values and fuzzy rules are formed on the base of data generated from C++ program. Figure 1 shows the fuzzy logic system which consists of two inputs: processing time and probability and three output variables namely, $H_2, H_3, H_4$. Figure 2 illustrates the membership functions for input processing time. Figure 3 illustrates the membership functions for input probability. Figure 4 illustrates the membership functions for output $H_5$. Figure 5 illustrates firing of the rule base.

5.1. Surfaces. The control surface for output $H_2$ is shown in Figure 6. The control surface for output $H_3$ is shown in Figure 7. The control surface for output $H_4$ is shown in Figure 8.

6. Numerical Illustration

Consider a 5-job, 4-machine sequencing problem whose processing times with their corresponding probabilities given in the Table 1.

Here, our objective is to obtain the latest time of the machines so that no machine remains idle.

Expected processing times are given in Table 2.

For the pair $(M_1, M_2)$, the in-out table is shown in Table 3 as

$$K_2 = t_{n,2} - \sum_{i=1}^{5} A_{i,2} = 49 - 34 = 15. \quad \text{(12)}$$

$H_2 = K_2 = 15$ units and also we have $H_1 = 0$.

For the pair $(M_2, M_3)$, the in-out table is shown in Table 4 as

$$K_3 = t_{n,3} - \sum_{i=1}^{5} A_{i,3} = 35 - 21 = 14, \quad \text{(13)}$$

$$H_3 = H_2 + K_3 = 15 + 14 = 29.$$
/* variable declaration */
static int job_prob[5][4]; // to store jobs time for machines
float float_exp_time[5][4]; // to store expected processing time
float m1m2[5][4], m2m3[5][4], m3m4[5][4];
float k2, k3, k4;
float h1, h2, h3, h4;

/* end variable declaration */

#define show_job_time_and_probability_for_machine()
#define exp_time_table()
#define prob_time_table()

void get_job_machine() // function to get jobs time and probability for machine
{
    int i, j;
    float arr2[5][4] = {0.2, 0.1, 0.2, 0.4, 0.2, 0.1, 0.2, 0.1, 0.1, 0.3, 0.2, 0.3, 0.3, 0.4, 0.2, 0.1, 0.2, 0.1, 0.2, 0.1, 0.3, 0.4, 0.2, 0.1, 0.2, 0.1, 0.2, 0.1};
    for (i = 0; i < 5; i++)
        for (j = 0; j < 4; j++)
            if (arr[i][j] > exp[arr2[i][j]])
                arr[i][j] = exp[arr2[i][j]];

    /* end of Put */
    printf(“

        \n Enter 5 Jobs Time & Probability for 4 Machines: “);

    for (i = 0; i < 5; i++)
        for (j = 0; j < 4; j++)
            scanf(“%d”, &job_machine[i][j]);

    printf(“

        \n Enter Job [ %d ] Time for Machine [ %d ]: “); #machine
        scanf(“%d”, &job_machine[i][j]);

    printf(“

        \n Enter Job [ %d ] Probability for Machine [ %d ]: “); #machine
        scanf(“%d”, &job_machine[i][j]);

    // end of get_job_machine function

void put_job_machine() // function to show jobs time and probability for machine
{
    int i, j;

    printf(“

        \n1 Machine1 | Machine2 | Machine3 | Machine4 | %n”);

    printf(“

        \nJobs | Time | Jobprob | Time | Jobprob | Time | Jobprob | Time | Jobprob

    “);

    for (i = 0; i < 5; i++)
        for (j = 0; j < 4; j++)
            printf(“%d|%.1f|%.1f|%.1f|%.1f
”, job_machine[i][j], job_prob[i][j], job_prob[i][j], job_prob[i][j], job_prob[i][j]);

    // end of Put_job_machine function

void get_exp_pro_time() // function to calculate expected processing time
{
    int i, j;
    for (i = 0; i < 5; i++)
        for (j = 0; j < 4; j++)
            exp_pro_time[i][j] = job_machine[i][j] * job_prob[i][j];

    // end of get_exp_pro_time function

void put_exp_pro_time() // function to showing expected processing time
{
    int i, j;

    printf(“

        \n Expected Processing Time: -
”);

    printf(“

        \n Jobs | Machine1 | Machine2 | Machine3 | Machine4
”);

    for (i = 0; i < 5; i++)
        for (j = 0; j < 4; j++)
            printf(“%d|%.1f
”, exp_pro_time[i][j]);

    // end of Put_exp_pro_time function

void get_in_out_table() // function to calculating In Out Table for machining
{
    int i, j, f, p1, p2;
    float machine1 = 0.0, machine2 = 0.0, machine3 = 0.0, machine4 = 0.0;

    m1m2[0][0] = 1; m1m2[0][1] = exp_pro_time[0][0] + p1 * p1 = m1m2[0][1];
    if (p2 > p1) m1m2[1][2] = p2; else m1m2[1][2] = p1;
    m1m2[2][3] = m1m2[2][2] + exp_pro_time[0][1];

    // Machine m2-m3 = 0.0; p2 = 0.0; for(i = 0; i < 5; i++)
    m2m3[0][0] = 1; m2m3[0][1] = exp_pro_time[0][1] + p1;
    p2 = m2m3[0][1];

    // Machine m3-m4 = 0.0; p2 = 0.0; for(i = 0; i < 5; i++)
    m2m3[0][0] = 1; m2m3[0][1] = exp_pro_time[0][1] + p1;
    p2 = m2m3[0][1];

    // end of get_in_out_table function

void put_in_out_table() // function to showing In Out Table for machining
{
    int i, j;

    printf(“

        \n In-Out Table: -
”);

    printf(“

        \n Jobs M1-M2 M2-M3 M3-M4
”);

Algorithm 1: Continued.
for(i=0;i<5;i++)
{ printf("%n
%d ",i+1);for(j=0;j<4;j++)
{ if(j==1 || j==3)
printf("-");else
printf(" ");
printf("%4.1f",m1m2[i][j]);if(j==3)
printf("%c",179);} 
for(j=0;j<4;j++)
{ if(j==1 || j==3)
printf("-");else
printf(" ");
printf("%4.1f",m2m3[i][j]);if(j==3)
printf("%c",179);} 
for(j=0;j<4;j++)
{ if(j==1 || j==3)
printf("-");else
printf(" ");
printf("%4.1f",m3m4[i][j]);if(j==3)
printf("%c",179);}}} // end of Put_in_out_table function //

void final()
{ int i;float sum=0.0;for(i=0;i<5;i++)sum=sum+exp_pro_time[i][1];
k2=m1m2[4][3];sum=sum;for(i=0;i<5;i++)
sum=sum+exp_pro_time[i][2];k3=m2m3[4][3];sum=sum;
for(i=0;i<5;i++)sum=sum+exp_pro_time[i][3];k4=m3m4[4][3];sum=sum;
h1=0;k2=h2;k3=h2+k3;k4=h3+k4;
printf("%n\nK2\n%.1f\nK3\n%.1f\nK4\n%.1f\nH1\n%.1f\nH2\n%.1f\nH3\n%.1f\nH4\n%.1f ");

void main()
{ clrscr();getch();get_job_machine();put_job_machine();getch();
get_exp_pro_time();put_exp_pro_time();getch();get_in_out_table();
put_in_out_table();getch();final();getch();
}

Algorithm 1: C++ program for given problem.

Figure 1
Table 1: The machines with processing time and corresponding probabilities.

<table>
<thead>
<tr>
<th>Jobs</th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
<th>$M_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>$a_{i,1}$</td>
<td>$p_{i,1}$</td>
<td>$a_{i,2}$</td>
<td>$p_{i,2}$</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>0.2</td>
<td>50</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0.2</td>
<td>40</td>
<td>0.1</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>0.1</td>
<td>30</td>
<td>0.3</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>0.3</td>
<td>20</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>0.2</td>
<td>80</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 2: Expected processing times.

<table>
<thead>
<tr>
<th>Jobs</th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
<th>$M_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>$A_{i,1}$</td>
<td>$A_{i,2}$</td>
<td>$A_{i,3}$</td>
<td>$A_{i,4}$</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>9</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>8</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3: The in-out table for machines $M_1$ and $M_2$.

<table>
<thead>
<tr>
<th>Jobs</th>
<th>$M_1$</th>
<th>$M_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>$A_{i,1}$</td>
<td>$A_{i,2}$</td>
</tr>
<tr>
<td>1</td>
<td>0–4</td>
<td>4–9</td>
</tr>
<tr>
<td>2</td>
<td>4–6</td>
<td>9–13</td>
</tr>
<tr>
<td>3</td>
<td>6–9</td>
<td>13–22</td>
</tr>
<tr>
<td>4</td>
<td>9–18</td>
<td>33–41</td>
</tr>
<tr>
<td>5</td>
<td>18–24</td>
<td>41–49</td>
</tr>
</tbody>
</table>

Table 4: The in-out table for machines $M_2$ and $M_3$.

<table>
<thead>
<tr>
<th>Jobs</th>
<th>$M_2$</th>
<th>$M_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>$A_{i,2}$</td>
<td>$A_{i,3}$</td>
</tr>
<tr>
<td>1</td>
<td>0–5</td>
<td>5–7</td>
</tr>
<tr>
<td>2</td>
<td>5–9</td>
<td>9–13</td>
</tr>
<tr>
<td>3</td>
<td>9–18</td>
<td>18–24</td>
</tr>
<tr>
<td>4</td>
<td>18–26</td>
<td>26–34</td>
</tr>
<tr>
<td>5</td>
<td>26–34</td>
<td>34–35</td>
</tr>
</tbody>
</table>
Table 5: The in-out table for machines $M_3$ and $M_4$.

<table>
<thead>
<tr>
<th>Jobs</th>
<th>$M_3$</th>
<th>$M_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_{i,3}$</td>
<td>$A_{i,4}$</td>
</tr>
<tr>
<td>1</td>
<td>0–2</td>
<td>2–14</td>
</tr>
<tr>
<td>2</td>
<td>2–6</td>
<td>14–22</td>
</tr>
<tr>
<td>3</td>
<td>6–12</td>
<td>22–25</td>
</tr>
<tr>
<td>4</td>
<td>12–20</td>
<td>25–27</td>
</tr>
<tr>
<td>5</td>
<td>20–21</td>
<td>27–29</td>
</tr>
</tbody>
</table>

For the pair $(M_3, M_4)$ the in-out table is shown in Table 5 as

\[
K_4 = t_{5,4} - \sum_{i=1}^{5} A_{i,4} = 2,
\]

\[
H_4 = H_3 + K_4 = 29 + 2 = 31.
\]

The in-out table for the machines with idle time zero is shown in Table 6.

<table>
<thead>
<tr>
<th>Jobs</th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
<th>$M_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_{i,1}$</td>
<td>$A_{i,2}$</td>
<td>$A_{i,3}$</td>
<td>$A_{i,4}$</td>
</tr>
<tr>
<td>1</td>
<td>0–4</td>
<td>15–20</td>
<td>29–31</td>
<td>31–43</td>
</tr>
<tr>
<td>2</td>
<td>4–6</td>
<td>20–24</td>
<td>31–35</td>
<td>43–51</td>
</tr>
<tr>
<td>3</td>
<td>6–9</td>
<td>24–33</td>
<td>35–41</td>
<td>51–54</td>
</tr>
<tr>
<td>4</td>
<td>9–18</td>
<td>33–41</td>
<td>41–49</td>
<td>54–56</td>
</tr>
<tr>
<td>5</td>
<td>18–24</td>
<td>41–49</td>
<td>49–50</td>
<td>56–58</td>
</tr>
</tbody>
</table>

Hence, we conclude that no machine remains idle.

7. Conclusion

The proposed algorithm provides the latest time at which processing of jobs on second, thired, and so on $n$th machine must be started such that these machines work continuously without any break until the last job is completed on them. The first machine has no idle time and, hence, works continuously; that is, the proposed algorithm helps the decision makers in determining the best latest time at which machines should be hired for a given set of jobs so as to minimize the total expected hiring cost with no-idle constraint. The study may be extended by introducing concepts of independent
setup time, transportation time, job block criteria, and non-availability constraints of machines for a certain interval of time.

References


