Hindawi Publishing Corporation
ISRN Operations Research
Volume 2013, Article ID 948541, 9 pages
http://dx.doi.org/10.1155/2013/948541

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. Ekel and X.-M. Yuan

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 Scharge [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 $n\times2$ 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 ith job on machine M_j,

$p_{i,j}$: probability associated to the processing time $a_{i,j}$,

$A_{i,j}$: expected processing time of ith job on machine M_j,

$t_{i,j}(S)$: completion time of ith job of sequence S on machine M_j,

$I_{i,j}(S)$: idle time of machine M_j for ith job in the sequence S,

$I^I_{i,j}(S)$: idle time of machine M_j for ith job in the sequence S when machine M_j starts at latest time L_j.

Definition 1. Completion time of ith job on machine M_j is denoted by $t_{i,j}$ and is defined as

$$\begin{aligned} t_{i,j} &= \max\left(t_{i-1,j}, t_{i,j-1}\right) + a_{i,j} \times p_{i,j} \quad \text{for } j \geq 2, \\ &= \max\left(t_{i-1,j}, t_{i,j-1}\right) + A_{i,j}, \end{aligned} \tag{1}$$

where $A_{i,j}$ = expected processing time of ith job on machine j.

Definition 2. Completion time of ith job on machine M_j when M_j starts processing jobs at time L_j is denoted by $t^I_{i,j}$ and is defined as

$$\begin{aligned} t^i_{i,j} &= L_J \sum_{K=1}^{I} A_{K,j} = \sum_{K=1}^{I} I_{k,j} + \sum_{K=1}^{I} A_{k,j}, \\ I_{i,j} &= \max\left(t_{i,j-1} - t_{i-1,j}, 0\right) \quad \text{for } j \geq 2. \end{aligned} \tag{2}$$

Also,

$$t^i_{i,j} = \max\left(t_{i,j}, t^i_{i-1,j}\right) + A_{i,j}. \tag{3}$$

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*

$$H_r = \max_{1\leq k\leq n}\left\{Y_k\right\}, \quad r = 2, 3, \ldots, m \tag{4}$$

where

$$\begin{aligned} Y_k &= t^l_{(k,r-1)} - \sum_{i=1}^{k-1} p_{i,r} X\, a_{ir} \quad \textit{for}\, k > 1, \\ Y_k &= t^l_{k,r-I} - \sum_{i=1}^{k-1} A_{ir}, \\ Y_1 &= t^l_{(1,r-1)}. \end{aligned} \tag{5}$$

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 $H_2 \geq t^l_{(1,1)}$ or

$$t^l_{(1,1)} \leq H_2. \tag{6}$$

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

$Y_q \geq Y_k$ for $k = 2, 3, \ldots, n$.

$Y_q \pm \sum_{i=1}^{k-1} A_{i,2} \geq Y_k + \sum_{i=1}^{k-1} A_{i,2}$

that is $H_2 + \sum_{i=1}^{k-1} A_{i,2} \geq t^l_{(k,1)} - \sum_{i=1}^{k-1} A_{i,2} + \sum_{i=1}^{k-1} A_{i,2}$

that is $t^l_{(k-1,2)} \geq t^l_{(k,1)}$ or

$$t^l_{(k,1)} \leq t^l_{(k-1,2)} \quad \text{for } k = 2, 3, \ldots, n. \tag{7}$$

Therefore, $I^l_{(k,2)} = \max[t^l_{(k,1)} - t^l_{(k-1,2)}, 0]$.

From (7),

$I^l_{(k,2)} = 0$ 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 $r = s + 1$.

But

$H_{s+1} = \max_{1\le k\le n}\{Y_k\}$.

Let $Y_t = \max_{1\le k\le n}\{Y_k\}$.

Therefore, $Y_t \ge Y_1$, that is, $H_{s+1} \ge t^l_{(s,1)}$

$$t^l_{(s,1)} \le H_{s+1}. \quad (8)$$

From (8), if machine M_{s+1} is taken on rent at time H_{s+1}, then it will start processing the first job without waiting. Therefore, idle time of machine M_{s+1} for the first job is zero when it starts processing jobs at time H_{s+1}.

For $k = 2, 3, \dots, n$.

$Y_t \ge Y_k$

which implies $Y_t + \sum_{i=1}^{k-1} A_{i,s+1} \ge Y_k + \sum_{i=1}^{k-1} A_{i,s+1}$

$H_{s+1} + \sum_{i=1}^{k-1} A_{(i,s+1)} \ge t^l_{(k,s)} - \sum_{i=1}^{k-1} A_{i,s+1} + \sum_{i=1}^{k-1} A_{i,s+1}$

which implies

$t^l_{(k-1,s+1)} \ge t^l_{(k,s)}$

$$t^l_{(k,s)} \le t^l_{(k-1,s+1)} \quad \text{for } k = 2, 3, \dots, n. \quad (9)$$

Therefore, $I^l_{(k,s+1)} = \max[t^l_{(k,s)} - t^l_{(k-1,s+1)}, 0]$

From (9),

$I^l_{(k,s+1)} = 0$ for $k = 2, 3, \dots, n$.

Therefore, the result is true for $r = s + 1$ also. □

Assumptions. (1) No machine processes more than one job at a time.

(2) Preemption of jobs is not allowed.

(3) Machines never breakdown during the scheduling process.

(4) Each job is processed through each of the machines once and only once.

(5) All the jobs and the machines are available at the beginning of the processing.

(6) Jobs are independent of each other.

4. Algorithm

The algorithm given in this paper provides the procedure to determine the times at which machines should be hired so that the idle time becomes zero, minimizing total expected hiring cost under the given policy.

Step 1. Calculate the expected processing time $A_{i,j} = a_{i,j}\times p_{i,j}$; for all $i, j = 1, 2, 3, \dots, m$.

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

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

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

$$H_j = H_{j-1} + K_j; \quad j = 3, 4, 5 \dots m-1,$$
$$H_2 = K_2, \qquad H_1 = 0. \quad (11)$$

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 based 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, and 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_3. 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

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

$H_2 = K_2 = 15$ 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

$$K_3 = t_{5,3} - \sum_{i=1}^{5} A_{i,3} = 35 - 21 = 14,$$
$$H_3 = H_2 + K_3 = 15 + 14 = 29. \quad (13)$$

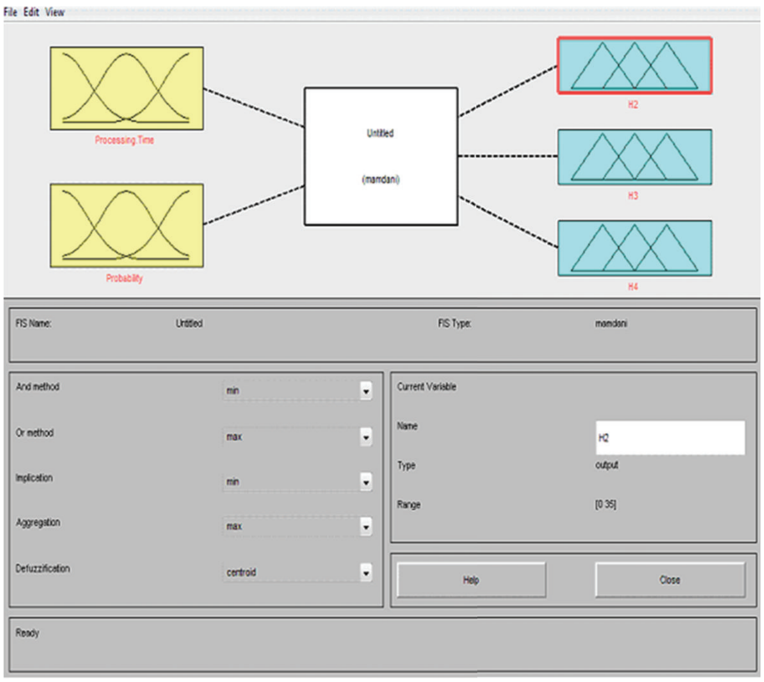
```
#include<stdio.h>
#include<conio.h>

// *************** variable declaration ***************//
static int job_machine[5][4]; //to store jobs time for machines
float job_prob[5][4]; // to tore jobs probability for machine
float exp_pro_time[5][4]; // to store expacted processing time
float m1m2[5][4],m2m3[5][4],m3m4[5][4];
float k2,k3,k4;
float h1,h2,h3,h4;
// ************** end variable declaration **************//
void get_job_machine() //function to get jobs time and probabiity for machine
{int i,j;/*
int arr[5][4]={10,40,5,20,25,30,20,40,   20,20,20,10, 30,5,25,40,   20,30,20,20};
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, };
  for(i=0;i<5;i++) { for(j=0;j<4;j++) { job_machine[i][j]=arr[i][j]; job_prob[i][j]=arr2[i][j];}   }
*/printf("\nEnter 5 Jobs Time & Probability for 4 Machines: ");
for (i=0;i<5;i++ ){for(j=0;j<4;j++){printf("\n\n Enter Job [ %d ] Time for Machine [ %d ]:
",i+1,j+1);scanf("%d",&job_machine[i][j]);
printf("\n\n Enter Job [ %d ] Probability for Machine [ %d ]: ",i+1,j+1);
scanf("%f",&job_prob[i][j]);}}}
// end of get_job_marchine function//
void put_job_machine() //function to show jobs time and probabiity for machine
{int i,j;
printf("\n\t   Machine1\t   Machine2\t   Machine3\t   Machine4\n");
printf("\nJobs\tTime\tProb.\tTime\tProb.\tTime\tProb.\tTime\tProb.");
for(i=0;i<5;i++)
{printf("\n\n[ %d ]",i+1);
for(j=0;j<4;j++){printf("\t%d\t%.1f",job_machine[i][j],job_prob[i][j]);}}}
// end of Put_job_marchine function//
void get_exp_pro_time() // function to calculatig 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\nExpected Proessing Time: -");
printf("\n\nJobs\t   Machine1\t   Machine2\t   Machine3\t   Machine4\n");for(i=0;i<5;i++)
{printf("\n\n[ %d ]",i+1);for(j=0;j<4;j++)
{printf("\t\t%.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 maching
{int i;float p1,p2;// Machine m1-m2p1=0.0;p2=0.0;for(i=0;i<5;i++)
{m1m2[i][0]=p1;m1m2[i][1]=exp_pro_time[i][0]+p1;p1=m1m2[i][1];
if(p2>p1)m1m2[i][2]=p2;elsem1m2[i][2]=p1;
m1m2[i][3]=m1m2[i][2]+exp_pro_time[i][1];p2=m1m2[i][3];}
// Machine m2-m3p1=0.0;p2=0.0;for(i=0;i<5;i++)
{m2m3[i][0]=p1;m2m3[i][1]=exp_pro_time[i][1]+p1;
p1=m2m3[i][1];if(p2>p1)m2m3[i][2]=p2;
elsem2m3[i][2]=p1;m2m3[i][3]=m2m3[i][2]+exp_pro_time[i][2];
p2=m2m3[i][3];}
// Machine m3-m4p1=0.0;p2=0.0;for(i=0;i<5;i++)
{m3m4[i][0]=p1;m3m4[i][1]=exp_pro_time[i][2]+p1;
p1=m3m4[i][1];if(p2>p1)m3m4[i][2]=p2;
elsem3m4[i][2]=p1;m3m4[i][3]=m3m4[i][2]+exp_pro_time[i][3];
p2=m3m4[i][3];}}
// end of get_in_out_table function//
void put_in_out_table() // function to showing In Out Table for maching
{int i,j;
printf("\n\nIn-Out Table: -");
printf("\n\nJobs M1-M2 M2-M3 M3-M4\n");
```

Algorithm 1: Continued.

```
for(i=0;i<5;i++)
{printf(“\n\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=0.0;
for(i=0;i<5;i++)sum=sum+exp_pro_time[i][2];k3=m2m3[4][3]-sum;
sum=0;for(i=0;i<5;i++)sum=sum+exp_pro_time[i][3];k4=m3m4[4][3]-sum;
h1=0;h2=k2;h3=h2+k3;h4=h3+k4;
printf(“\n\nK2\tK3\tK4\tH1\tH2\tH3\tH4”);
printf(“\n\n%.1f\t%.1f\t%.1f\t%.1f\t%.1f\t%.1f\t%.1f”,k2,k3,k4,h1,h2,h3,h4);}
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.



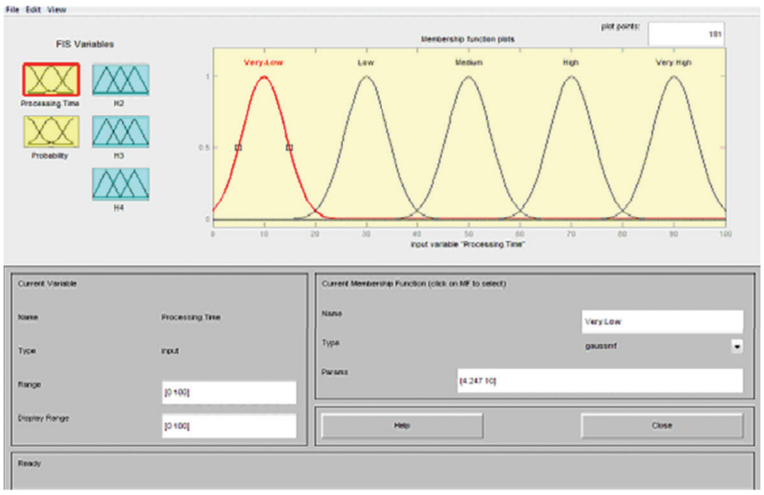
FIS Name: Untitled
FIS Type: mamdani

And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

Current Variable
Name: H2
Type: output
Range: [0 35]

Ready

Figure 1

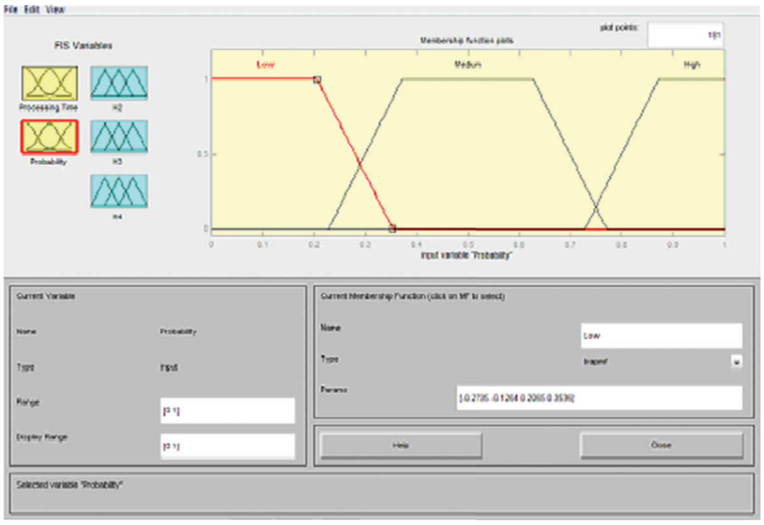


Current Variable
Name Processing Time
Type input
Range [0 100]
Display Range [0 100]

Current Membership Function (click on MF to select)
Name Very Low
Type gaussmf
Params [4.247 10]

Ready

Figure 2

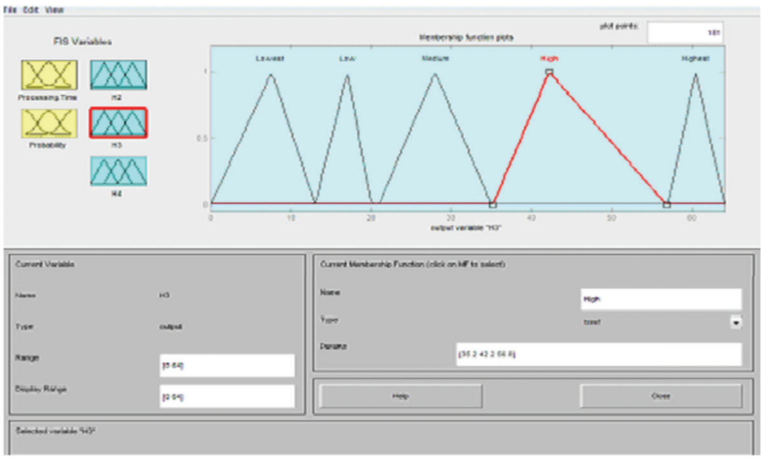


Current Variable
Name Probability
Type input
Range [0 1]
Display Range [0 1]

Current Membership Function (click on MF to select)
Name Low
Type trapmf
Params [-0.2735 -0.1264 0.2065 0.3536]

Selected variable "Probability"

Figure 3

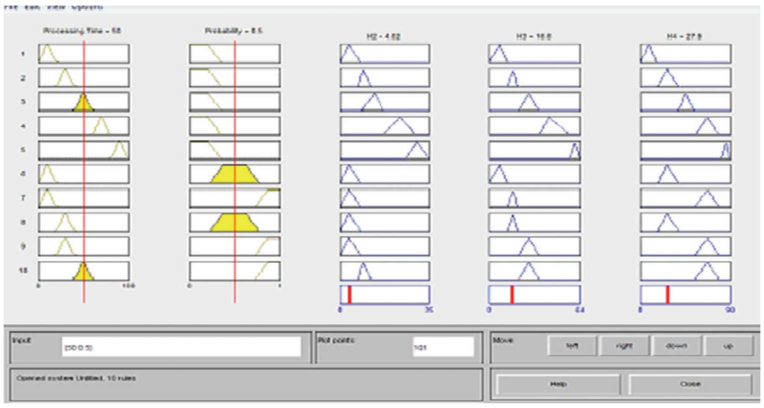


Current Variable
Name H3
Type output
Range [5 64]
Display Range [2 64]

Current Membership Function (click on MF to select)
Name High
Type trimf
Params [35.2 42.2 56.8]

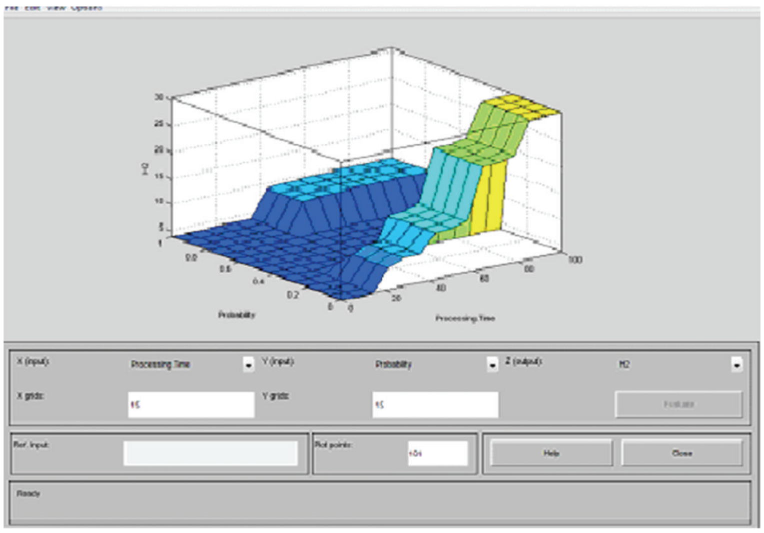
Selected variable "H3"

Figure 4



Input: [50 0.5]
Plot points: 101
Opened system Untitled, 10 rules

Figure 5



X (input): Processing Time
Y (input): Probability
Z (output): H2
X grids: 15
Y grids: 15
Plot points: 101
Ready

Figure 6

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

Jobs	M_1		M_2		M_3		M_4	
i	$a_{i,1}$	$p_{i,1}$	$a_{i,2}$	$p_{i,2}$	$a_{i,3}$	$p_{i,3}$	$a_{i,4}$	$p_{i,4}$
1	20	0.2	50	0.1	10	0.2	30	0.4
2	10	0.2	40	0.1	20	0.2	80	0.1
3	30	0.1	30	0.3	30	0.2	10	0.3
4	30	0.3	20	0.4	40	0.2	20	0.1
5	30	0.2	80	0.1	5	0.2	20	0.1

Table 2: Expected processing times.

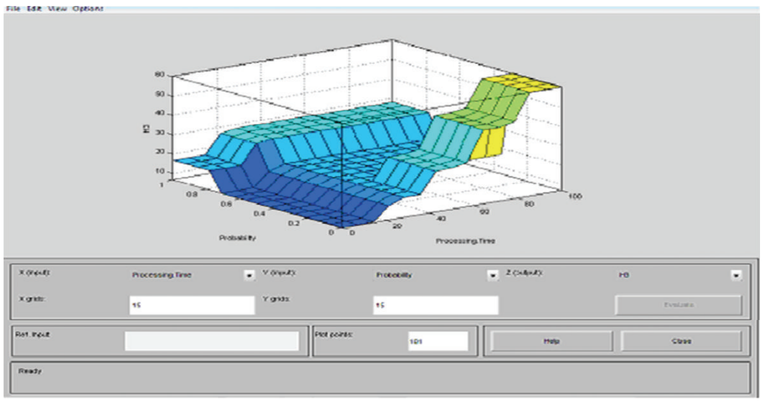
Jobs	M_1	M_2	M_3	M_4
i	$A_{i,1}$	$A_{i,2}$	$A_{i,3}$	$A_{i,4}$
1	4	5	2	12
2	2	4	4	8
3	3	9	6	3
4	9	8	8	2
5	6	8	1	2

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

Jobs	M_1	M_2
	$A_{i,1}$	$A_{i,2}$
1	0–4	4–9
2	4–6	9–13
3	6–9	13–22
4	9–18	33–41
5	18–24	41–49

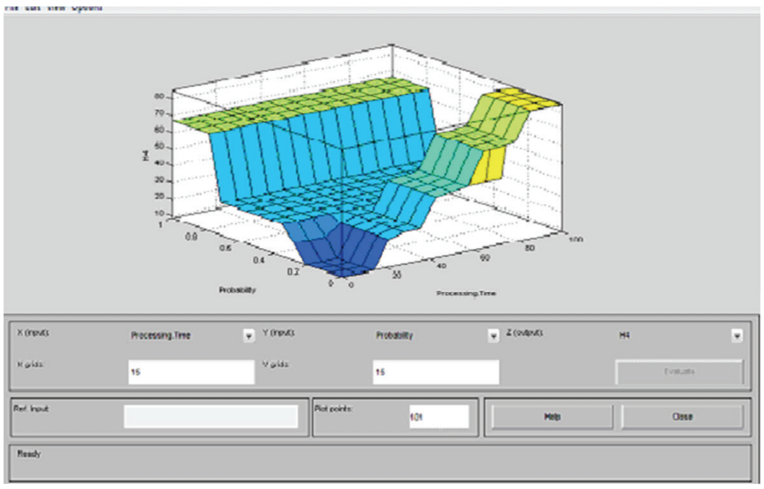
Table 4: The in-out table for machines M_2 and M_3.

Jobs	M_2	M_3
i	$A_{i,2}$	$A_{i,3}$
1	0–5	5–7
2	5–9	9–13
3	9–18	18–24
4	18–26	26–34
5	26–34	34-35



X (input): Processing Time
Y (input): Probability
Z (output): H3
X grids: 15
Y grids: 15
Ref. Input:
Plot points: 101
Ready

Figure 7



X (input): Processing Time
Y (input): Probability
Z (output): H4
X grids: 15
Y grids: 15
Ref. Input:
Plot points: 101
Ready

Figure 8

Table 5: The in-out table for machines M_3 and M_4.

Jobs	M_3 $A_{i,3}$	M_4 $A_{i,4}$
1	0–2	2–14
2	2–6	14–22
3	6–12	22–25
4	12–20	25–27
5	20-21	27–29

Table 6: In-out table.

Jobs	M_1 $A_{i,1}$	M_2 $A_{i,2}$	M_3 $A_{i,3}$	M_4 $A_{i,4}$
1	0–4	15–20	29–31	31–43
2	4–6	20–24	31–35	43–51
3	6–9	24–33	35–41	51–54
4	9–18	33–41	41–49	54–56
5	18–24	41–49	49-50	56–58

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

$$K_4 = t_{5,4} - \sum_{i=1}^{5} A_{i,4} = 2, \qquad H_4 = H_3 + K_4 = 29 + 2 = 31. \tag{14}$$

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

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 mth 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

[1] S. M. Johnson, "Optimal two and three stage production schedule with setup times included," *Naval Research Logistics Quarterly*, vol. 1, pp. 61–68, 1954.

[2] E. Ignall and L. Scharge, "Application of branch and bound technique to some flow shop scheduling problems," *Operation Research*, vol. 13, pp. 400–412, 1965.

[3] I. Adiri and D. Pohoryles, "Flowshop/no-idle or no-wait scheduling to minimize the sum of completion times," *Naval Research Logistics Quarterly*, vol. 29, no. 3, pp. 495–504, 1982.

[4] C. Rajendran and D. Chaudhuri, "An efficient heuristic approach to the scheduling of jobs in a flowshop," *European Journal of Operational Research*, vol. 61, no. 3, pp. 318–325, 1992.

[5] W. Szwarc, "Special cases of the flow-shop problem," *Naval Research Logistics Quarterly*, vol. 24, no. 3, pp. 483–492, 1977.

[6] T. Yoshida and K. Hitomi, "Optimal two-stage production scheduling with setup times separated," *AIIE Transactions*, vol. 11, no. 3, pp. 261–263, 1979.

[7] Anup, "On two machine flow shop problem in which processing time assume probabilities and there exists equivalent for an ordered job bloc," *Journal of the Indian Society of Statistics and Operations Resear*, vol. 23, pp. 41–44, 2002.

[8] T. P. Singh, K. Rajindra, and G. Deepak, "Optimal three stage production schedule the processing time and set up times associated with probabilities including job block criteria," in *Proceedings of National Conference FACM*, pp. 463–470, 2005.

[9] D. Gupta, T. P. Singh, and R. Kumar, "Minimizing rental cost in two stage flow shop, the processing time associated with probabilities including job block," *Reflections de ERA*, vol. 1, no. 2, pp. 107–120, 2006.

[10] D. Gupta and H. Singh, "The idle/waiting time operator with applications to multistage flow shop scheduling to minimize the rental cost under specified rental policy where processing times are associated with probabilities including transportation time," *Industrial Engineering Letters*, vol. 2, no. 3, pp. 61–70, 2012.

[11] L. Narain and P. C. Bagga, "Minimizing total elapsed time subject to zero total idle time of machines in n $\times$ 3 flowshop problem," *Indian Journal of Pure and Applied Mathematics*, vol. 34, no. 2, pp. 219–228, 2003.

[12] L. Narain and P. C. Bagga, "Flowshop/no-idle scheduling to minimise the mean flowtime," *ANZIAM Journal*, vol. 47, no. 2, pp. 265–275, 2005.

[13] L. Narian and P. C. Bagga, "Scheduling problems in rental situation," *Bulletin of Pure and Applied Sciences: Section E*, vol. 24, 2005.