Innovative Systems Design and Engineering www.iiste.org
ISSN 2222-1727 (Paper) ISSN 2222-2871 (Online)
Vol 2, No 4, 2011

Idle / Waiting Time Operator $O_{i,w}$ Of An Equivalent Job For A Job Block Criteria To Minimize The Rental Cost In Two Stage Flow Shop Scheduling, the Processing Time Associated With Probabilities

Deepak Gupta
Prof. & Head, Dept. of Mathematics,
Maharishi Markandeshwar University, Mullana, Ambala, India
Email: guptadeepak2003@yahoo.co.in

Sameer Sharma (Corresponding Author)
Assistant Professor, Dept. of Mathematics,
D.A.V. College, Jalandhar, Punjab, India
, Email: samsharma31@yahoo.com

Naveen Gulati
Assistant Professor, Dept. of Mathematics,
S.D.College, Ambala Cantt , Haryana, India
Email: naveengulatimaths@gmail.com

Harminder Singh
Research Scholar, Dept. of Mathematics
Maharishi Markandeshwar University, Mullana, Ambala, India
Email: harminder.cheema85@gmail.com

Abstract

The present paper is an attempt to study the n x 2 flow shop production scheduling problem in which the processing times are associated with their respective probabilities and follows some restrictive rental policy including equivalent job block criteria. The objective of the study is to get optimal sequence of the jobs in order to minimize the rental cost using idle/waiting time operator through the iterative algorithm. The operator technique is an easy approach in economical and computational point of view and gives an optimal schedule rule in order to minimize the rental cost of the machines. The model is justified by a computer programme followed by a numerical illustration.

Keywords: Equivalent–job, rental policy, makespan, elapsed time, idle time, Idle/waiting time operator $O_{i,w}$ etc.

2000 Mathematics Subject Classification: Primary: 90B35; Secondary: 90B30.

1. Introduction

Many practical and industrial situations which generally arise in production concern to get an optimal schedule of jobs in a set of machines diverted the attention of researchers, engineers and academic community. Various techniques have been search out to attempt multistage flow shop scheduling problem such as critical path method, branch and bound algorithm, heuristic method, Gants Charts, method of adjacent pair wise job interchange etc. The optimization algorithm for two, three multistage flow ship problem in order to minimize the processing times have been developed by Johnson (1954), Ignall and Scharge (1965), Campbell (1970), Maggu and Das (1977), Szwarc W. (1977), Singh T.P. and Deepak Gupta (1985, 2005), Bagga P.C. et al. (2003) by considering various parameters. Maggu and Das(1977,1980) introduced the concept of equivalent job block criteria for a job block in the theory of scheduling and idle/waiting time operator. Singh T.P. and Gupta Deepak et al. (2005) studied the optimal three stage flow shop problem in which the processing time and set up time both are associated with probabilities including the job block criteria. Gupta Deepak and Sharma Sameer (2011) studied the minimization of rental cost under specified rental policy in two stage flow shop, the processing time associated with probabilities including break-down interval and Job – block criteria. The heuristic approach has been adopted in all.

In flow-shop scheduling, the objective is to obtain a sequence of jobs which when processed in a fixed order of machines, will optimize some well defined criteria. The concept of equivalent-job block in the theory of scheduling is useful and significant in the sense to create a balance between the cost of providing priority in service to the customer and cost of giving services with non priority customers. The decision maker may decide how much to charge extra from the priority customer. *Singh T.P., Gupta Deepak* [2006] studied *n* x 2 general flow shop problem to minimize rental cost under a pre-defined rental policy in which the probabilities have been associated with processing time on each machine including job block criteria. We have extended the study made by *Singh T.P., Gupta Deepak* by introducing the application of idle waiting time operator $O_{i,w}$ as defined by Maggu and Das (1980) in scheduling theory. The paper differs from Maggu and Das (1980) in the sense that here the probabilities are associated with processing time on each machine. The operator technique is an easy approach in economical and computational point of view as in comparison to the heuristic approach. We have developed an algorithm minimizing the utilization time of second machine combined with Johnson's algorithm in order to minimize the rental cost of the machines.

Theorem 1 *Let n jobs 1, 2, 3, ……n are processed through two machines A & B in order AB with processing time* a_i & b_i*(i = 1, 2, 3, ……n) on machine A and B respectively.*

If $(a_p, b_p)\ O_{i,w}\ (a_q, b_q) = (a_\beta, b_\beta)$

then $a_\beta = a_p + max\ (a_q - b_p, 0)$

and $b_\beta = b_q + max\ (b_q - a_q, 0)$

where β is the equivalent job for job block (p, q) and p, q ϵ {1, 2, 3, ……n}.

Proof*:* Starting by the equivalent job block criteria theorem for β = (p, q) given by Maggu & Das [6], we have:

$a_\beta = a_p + a_q - \min (b_p, a_q)$ …(1)

$b_\beta = b_p + b_q \min (b_p, a_q)$ …(2)

Now, we prove the above theorem by a simple logic:

Case I: When $a_q > b_p$

$a_q > b_p > 0$

$\max \{ a_q > b_p, 0\} = a_q > b_p$ …(3)

and

$b_p > a_q < 0$

$\max \{ b_p > a_q, 0\} = 0$ …(4)

(1) $a_\beta = a_p + a_q - \min (b_p, a_q)$

$= a_p + a_q - b_p$ as $a_q > b_p$

$= a_p + \max \{a_q - b_p, 0\}$ (using (3) …(5)

(2) $b_\beta = b_p + b_q - \min (b_p, a_q)$

$= b_p + b_q - b_p$ as $a_q > b_p$

$= b_q + (b_p - b_p)$

$= b_q + 0$

$= b_q + \max (b_p - a_q, 0)$ (using (4) … (6)

Case II: When $a_q < b_p$

$a_q - b_p < 0$

$\max (a_q - b_p, 0) = 0$ … (7)

and $b_p - a_q > 0$

$\max (b_p - a_q, 0) = b_p - a_q$ …(8)

(1) $a_\beta = a_p + a_q - \min (b_p, a_q)$

$= a_p + a_q - a_q$ as $a_q > b_p$

$= a_p + 0$

$= a_p + \max(a_q - b_p, 0)$ (using (7)) …(9)

(2) $b_\beta = b_p + b_q - \min (b_p, a_q)$

$= b_p + b_q - a_q$ as $a_q < b_p$

$= b_p + (b_p - a_q)$

$= b_p + \max (b_p - a_q, 0)$ (using (8)) … (10)

Case III: When $a_q = b_p$,$a_q - b_p = 0$

Therefore, $\max (a_q - b_p, 0) = 0$ … (11)

Also $b_p - a_q = 0$

Therefore, $\max (b_p - a_q, 0) = 0$ … (12)

(1) $a_\beta = a_p + a_q - \min (b_p, a_q)$

$= b_p + a_q - a_p$ as $b_q = a_p$

$= a_p + 0$

$= a_p + \max (a_q - b_p, 0)$ … (13)

(2) $b_\beta = b_p + b_q - \min (b_p, a_q)$

$= b_p + b_q - b_p$

$= b_q + (b_p - b_p)$

$= b_q + 0$

$= b_q + \max (b_p - a_q, 0)$ (using (12) …(14)

By (5), (6), (9), (10), (13) and (14) we conclude:

$a_\beta = a_p + \max (a_q - b_p, 0)$

$b_\beta = b_p + \max (b_p - a_q, 0)$ for all possible three cases.

The theorem can be generalized for more number of job blocks as stated:

Let n jobs 1, 2, 3, ……..n are processed through two machines A & B in order AB with processing time a_i & b_i (i = 1, 2, 3, ……..n) on machine A & B respectively.

If $(a_{io}, b_{io})\ O_{i,w}\ (ai_1, bi_1)\ O_{i,w}\ (ai_2, bi_2)\ O_{i,w}\ldots\ldots\ldots\ldots\ldots\ O_{i,w}\ (a_{ip}, b_{ip}) = (a_\beta, b_\beta)$

Then

$$a_\beta = a_{io} + \sum_{J=1}^{p} \max\{a_{ij} - b_{i(j-1)}\ 0\}$$

and $b_\beta = b_{ip} + \sum_{J=1}^{p} \max\{b_{i(j-1)} - a_{ij}, 0\}$

where $i_0, i_1, i_2, i_3, \ldots\ldots\ldots\ldots ip \in \{1, 2, 3\ \ldots\ldots\ldots\ldots n\}$ and β is the equivalent job for job block $(i_0, i_1, i_2, i_3, \ldots\ldots\ldots\ldots ip)$. The proof can be made using Mathematical induction technique on the lines of Maggu & Das [8].

In the light of above theorem operator $O_{i,w}$ (Idle/Waiting time Operator) is defined as follows:

Definition 1

Let R_+ be the set of non negative numbers. Let $G = R_+ \times R_+$. Then $O_{i,w}$ is defined as a mapping from G x G → G given by:

$$O_{i,w}[(x_1, y_1), (x_2, y_2)] = (x_1, y_1)\ O_{i,w}\ (x_2, y_2)$$
$$= [x_1 + \max((x_2 - y_1), 0), y_2 + \max((y_1 - x_2), 0)], \text{ where } x_1, x_2, y_1, y_2 \in R_+.$$

Definition 2

An operation is defined as a specific job on a particular machine.

Definition 3

Total elapsed time for a given sequence

= Sum of expected processing time on 2 nd machine (M_2) + Total idle time on M_2

$$= \sum_{i=1}^{n} B'_i + \sum_{i=1}^{n} I_{i2} = \sum_{i=1}^{n} B'_i + \max[P_k], \text{ where } P_K = \sum_{i=1}^{k} A'_i - \sum_{i=1}^{k-1} B'_i .$$

2. Practical Situation

Various practical situations 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. In his starting career, we find a medical practitioner does not buy expensive machines say X-ray machine, the Ultra Sound Machine, Rotating Triple Head Single Positron Emission Computed Tomography Scanner, Patient Monitoring Equipment, and Laboratory Equipment etc., but instead takes on rent. Rental of medical equipment is an affordable and quick solution for hospitals, nursing homes, physicians, which are presently constrained by the availability of limited funds due to the recent global economic recession. Renting enables saving working capital, gives option for having the equipment, and allows upgradation to new technology.

Sometimes the priority of one job over the other is preferred. It may be because of urgency or demand of its relative importance, the job block criteria becomes important.

3. Assumptions

1. Machine break down is not considered. This simplifies the problem by ignoring the stochastic component of the problem.

2. Jobs are independent to each other.

3. We assume rental policy that all the machines are taken on rent as and when they are required and are returned as when they are no longer required for processing. Under this policy second machine is taken on rent at time when first job completes its processing on first machine. Therefore idle time of second machine for first job is zero.

4. Pre- emption is not allowed i.e. jobs are not being split, clearly, once a job started on a machine, the process on that machine can't be stopped unless the job is completed

4. Notations

S: Sequence of jobs 1,2,3,….,n

M_j: Machine j, j= 1,2,…….

A_i : Processing time of i^{th} job on machine A.

B_i: Processing time of i^{th} job on machine B.

A_i' : Expected processing time of i^{th} job on machine A.

B_i' : Expected processing time of i^{th} job on machine B.

p_i : Probability associated to the processing time A_i of i^{th} job on machine A.

q_i : Probability associated to the processing time B_i of i^{th} job on machine B.

β : Equivalent job for job – block.

S_i : Sequence obtained from Johnson's procedure to minimize rental cost.

C_j : Rental cost per unit time of machine j.

U_i : Utilization time of B (2 nd machine) for each sequence S_i

$t_1(S_i)$: Completion time of last job of sequence S_i on machine A.

$t_2(S_i)$: Completion time of last job of sequence S_i on machine B.

$R(S_i)$: Total rental cost for sequence S_i of all machines.

$CT(S_i)$:Completion time of 1^{st} job of each sequence S_i on machine A.

5. Problem Formulation

Let n jobs 1, 2, 3,……,n be processed through two machines A and B ,with no passing allowed. Let a_{ij} denotes the processing time of i^{th} job on j^{th} machine with their respective probabilities p_{ij} such that $\sum p_{ij} = 1$. Let $\beta = (l, m)$ be an equivalent job for job block in which job l is given priority on a job m. Also we consider the following structure relationship holds good.

Either $\min (a_{il}, p_{il}) \geq \max (a_{ij}\ p_{ij})$ for j = 2, 3 … m-1.

Or $\min (a_{im}\ p_{im}) \geq \max (a_{ij}\ p_{ij})$ for j = 2, 3… m-1.

The mathematical model of the problem can be tabulated as in **table1.**

Rental Policy (P)

The machines will be taken on rent as and when they are required and are returned as and when they are no longer required.

Mathematically, the problem is stated as:

Minimize $t_2(S_k)$

and Minimize $R(S_k) = t_1(S_k) \times C_1 + U_k \times C_2$

Subject to the constraint Rental policy P.

6. Algorithm

Based on the above theorem and equivalent job block theorem by Maggu & Das the algorithm in a modified form to minimize the total rental cost of machines can be depicted as below:

Step 1: Define expected processing time A_i' & B_i' on machine A & B respectively as follows:

$A_i' = A_i \times p_i$

$B_i^{'} = B_i \times q_i$

Step 2: Determine equivalent jobs for each job block using operator theorem and concept of the idle/waiting time operator $O_{i,w}$ as per definition.

Step 3: Using Johnson's two machine algorithm obtain the sequence S_1 , while minimize the total elapsed time.

Step 4: Observe the processing time of 1 st job of S_1 on the first machine A. Let it be α.

Step 5: Obtain all the jobs having processing time on A greater than α. Put these job one by one in the 1 st position of the sequence S_1 the same order. Let these sequences be S_2, S_3, S_4,…S_r

Step 6: Prepare in-out table for each sequence S_i (i = 1,2,…r) and evaluate total completion time of last job of each sequence, .i.e $t_1(S_i)$ and $t_2(S_i)$ on machine A & B respectively.

Step 7: Evaluate completion time $CT(S_i)$ of 1 st job of each sequence S_i on machine A.

Step 8: Calculate utilization time U_i of 2 nd machine for each sequence S_i as:

$U_i = t_2(S_i) - CT(S_i)$ for i=1,2,3,…r.

Step 9: Find Min $\{U_i\}$, i=1,2,…r. let it be corresponding to i = m, then S_m is the optimal sequence for minimum rental cost.

Min rental cost = $t_1(S_m) \times C_1 + U_m \times C_2$

Where C_1 & C_2 are the rental cost per unit time of 1st & 2 nd machine respectively.

7.Programme

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<string.h>
#define ROWS 5

float table[5][2];
int r_costA,r_costB;
int p,q;
float beta_left,beta_right;
float  ap,aq,bp,bq;
float max1,max2;
float tab[4][2];
int i,j,c;
float min;
char ign_row[10];
int t=0;
int ign_index=0;
char str_left[10];
char str_right[10];
int index1=0, index2=0;
int flag=0, l=0,last;
```

```
char s1[10], s2[10], s3[10], s4[10];
int s=0;
int index[5];
float a_in[5],a_out[5], b_in[5],b_out[5];
float aout,bout;
float a1_uti[5], b1_uti[5];
float atime,btime;
void input_data(void)
{
    float p_time,prob;
    int i;
    clrscr();
for(i=0;i<ROWS;i++)
{
    printf("Enter The Processing Time of Job [ %d ] for Maching A : ",i+1);
    scanf("%f",&p_time);
    printf("Enter The Probability of job [ %d ] for Machine A    : ",i+1);
    scanf("%f",&prob);
    table[i][0]=p_time*prob;
    printf("Enter The Processing Time of Job [ %d ] for Maching B : ",i+1);
    scanf("%f",&p_time);
    printf("Enter The Probability of job [ %d ] for Machine B    : ",i+1);
    scanf("%f",&prob);
    table[i][1]=p_time*prob;
}
    printf("Enter Rental Cost for Machine A : ");
    scanf("%d",&r_costA);
    printf("Enter Rental Cost for Machine B : ");
    scanf("%d",&r_costB);
    printf("Enter Job Blocks : ");
    scanf("%d",&p);
    scanf("%d",&q);
}
void show_table(void)
{
    printf("\nJobs\tMachine A\tMachine B\n");
for(i=0;i<5;i++)
{
```

```
        printf("\n%d\t%f\t%f",i+1,table[i][0],table[i][1]);
    }
    }
    void calculate_beta(void)
    {
      // [-------ap-------------] +
    //          [-------p--]

        ap=table[p-1][0];
        aq=table[q-1][0];
        bp=table[p-1][1];
        bq=table[q-1][1];
        printf("\nap %f\t aq %f",ap,aq);
        printf("\nbp %f\t bq %f",bp,bq);
        max1=aq-bp;
    if(max1>0)
        beta_left=ap+max1;
    else
        beta_left=ap;
        printf("\n Beta left %f",beta_left);
        max2=bp-aq;
    if(max2>0)
        beta_right=bq+max2;
    else
        beta_right=bq;
        printf("\n Beta right %f\n",beta_right);
    // after calculating beta//
    //making table //
        tab[p-1][0]=beta_left;
        tab[p-1][1]=beta_right;
        for(i=0;i<5;i++)
    {
    if(i==q-1 || i==p-1)
        continue;
    for(j=0;j<2;j++)
    {
        tab[i][j]=table[i][j];
    }
```

```
}
}//end of function calculate betta
// end of function show_tab1-------------------------------------------
void show_tab1(void)
{
for(i=0;i<5;i++)
{
    if(i==q-1)
    continue;
    printf("\n");
    for(j=0;j<2;j++)
    {
            printf("\t%d\t%f",i+1,tab[i][j]);
    }
}
}
// end of function show_tab1 -------------------------------------------
    int ignore_row(int n)
{
    int in=0;
    while(ign_row[in]!=NULL)
{
if(n==(ign_row[in]-48))
    return 1;
    in++;
}
return 0;
}
/// ----------------------------------------------------------------
void set_min(void)
{

    for(i=0;i<5;i++)
    {
            flag=0;
            if(i==q-1 || ignore_row(i))
            flag=1;
            if(flag==1)
```

```
                continue;
                min=tab[i][0];
                return;
        }
}
void locate_min()
{
        for(i=0;i<5;i++)
        {
                if(i==q-1 || ignore_row(i))
                continue;
                for(j=0;j<2;j++)
                {
                if(min>tab[i][j])
                min=tab[i][j];
                }
        }
}
// function to set_min value ----------------------------------------
void get_postion_min()
{
        for(i=0;i<5;i++)
        {
                if(i==q-1 || ignore_row(i))
                continue;
                for(j=0;j<2;j++)
                {
                        if(min==tab[i][j])
                        break;
                }
                if(j<2)
                break;
        }
        if(i==p-1)
        {
                if(j==0)
                {
                        str_left[index1++]=p+48;
```

```
                    str_left[index1++]=q+48;
                }
                else
                {
                str_right[index2++]=q+48;
                    str_right[index2++]=p+48;
                }
        }
        else
        {
                if(j==0)
                {
                    str_left[index1++]=i+49;
                }
                else
                {
                    str_right[index2++]=i+49;
                }
        }
        ign_row[t++]=i+48;
        ign_row[t]=NULL;
        str_left[index1]=NULL;
        str_right[index2]=NULL;
    }//end of function geting position of minimum value
    /// function to create sequence
    void create_sec_string(char *st)
    {
        if(s==0)
        {
        strcpy(st,str_left);
        strcat(st,str_right);
        s++;
        }
        else    //else 1
        {
                int len;
                int temp;
                t=0;
```

```
strcpy(st,s1);
if((st[s]-48)==q)
{
s++;
}
if( (st[s]-48)==p)
{
s++;
t=s;
        while(t>0)
        {
                temp=st[t];
                st[t]=st[t-1];
                st[t-1]=temp;
                t--;
        }
        t=s;
        while(t>0)
        {
        temp=st[t];
        st[t]=st[t-1];
        st[t-1]=temp;
        t--;
        }
}
else
{
t=s;
        while(t>0)
        {
        temp=st[t];
        st[t]=st[t-1];
        st[t-1]=temp;
        t--;
        }
}
 s++;
}//else1
```

```
        printf("\n Sequence  :%s",st);
}
// end of creat sequence function----------------------------------------
//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
void get_in_out(char *st,int n)
{
for(i=0;i<5;i++)
{
     a_in[i]=0;
     a_out[i]=0;
     b_in[i]=0;
     b_out[i]=0;
}
     aout=0;
     bout=0;
for(i=0;st[i]!=NULL;i++)//for 1
     {
     j=0;
     j=st[i]-48;
     j--;
                if(i==0)
                {
                        index[i]=j+1;
                        a_in[i]=0.0f;
                  aout=a_out[i]=table[j][0];
                        b_in[i]=aout;
                  bout=b_out[i]=aout+table[j][1];
                }
                else
                {
                        index[i]=j+1;
                        a_in[i]=aout;
                  aout=a_out[i]=a_in[i]+table[j][0];

                        if(aout>bout)
                        {
                                b_in[i]=aout;
                          bout=b_out[i]=aout+table[j][1];
```

```
                }
                else
                {
                        b_in[i]=bout;
                  bout=b_out[i]=bout+table[j][1];
                }
            }
        }//end of for 1
        a1_uti[n-1]=a_out[i-1];
        b1_uti[n-1]=b_out[i-1]-a_out[0];
    }
    //||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    // function to  show in out ---------
    void show_in_out()
    {
        for(i=0;i<5;i++)
        {
        printf("\n%d\t%f\t%f\t%f\t%f",index[i],a_in[i],a_out[i],b_in[i],b_out[i]);
        }
    }
    // function end show in out
    //||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
    void get_min_uti_time()
    {     btime=b1_uti[0];
        atime=a1_uti[0];
        for(i=1;i<4;i++)
        {
          if(btime>b1_uti[i])
          {
          btime=b1_uti[i];
          atime=a1_uti[i];
          }
        }
        printf("\n Machine a minimum uti time :%f",atime);
        printf("\n Machine b minimum uti time :%f",btime);
    }
    void all_utime()
    {
```

```
for(i=0;i<5;i++)
{
    printf("\n%f\t%f",a1_uti[i],b1_uti[i]);
}
}
void main()
{
    int lop=0;
    clrscr();
    ign_row[0]=NULL;
    input_data();
    show_table();
    min=tab[0][0];
    calculate_beta();
    show_tab1();
for(lop=0;lop<4;lop++)
{
    set_min();
    locate_min();
    get_postion_min();
}
    printf("\n left :%s",str_left);
    printf("\n Right :%s",strrev(str_right));
    printf("\nPress any key to Conti...");
    getch();
//-------------------------------------------------
    create_sec_string(s1);
    create_sec_string(s2);
    create_sec_string(s3);
    create_sec_string(s4);
    printf("\nPress any key to Conti...");
    getch();
//-------------------------------------------------
    show_table();
    printf("\n Sequence 1 : %s",s1);
    printf("\n ------------------------------------------------------------:\n");
    get_in_out(s1,1);
    show_in_out();
```

```
        printf("\n Sequence 2 : %s",s2);
        printf("\n ----------:\n");
        get_in_out(s2,2);
        show_in_out();
        printf("\n Sequence 3 : %s",s3);
        printf("\n ----------:\n");
        get_in_out(s3,3);
        show_in_out();
        printf("\n Sequence 4 : %s",s4);
        printf("\n ----------:\n");
        get_in_out(s4,4);
        show_in_out();
    //---------------------------------------------------------------------
        all_utime();
        get_min_uti_time();
        printf("\n Total Rental Cost is :%f",atime*r_costA+btime*r_costB);
        getch();
}.
```

8. Numerical Illustration

Consider 5 jobs and 2 machines problem to minimize the rental cost. The processing times with their respective associated probabilities are given as follows. Obtain the optimal sequence of jobs and minimum rental cost of the complete set up, given rental costs per unit time for machines M_1 & M_2 are 21 and 17 units respectively, and jobs (1, 3) are to be processed as an equivalent group jobs.

Jobs	**Machine A**		**Machine B**	
	A_i	**p_i**	**B_i**	**q_i**
1	2 5	0 .2	1 3	0 .3
2	3 3	0 .1	2 1	0 .2
3	1 5	0 .3	1 9	0 .1
4	1 7	0 .2	2 3	0 .1
5	1 1	0 .2	1 7	0 .3

(Table-2)

Solution: As per Step 1: Expected processing time are as shown in **table 3**.

As per Step 2: The expected processing time for the equivalent job on fictitious machine are shown in **table 4**.Here we have

$(a_p, b_q)\ O_{i,w}\ (a_q, b_q) = (a_\beta, b_\beta)$

$= \{(a_p + \max(a_q - b_p, 0), (b_q + \max(b_p - a_q, 0)\}$

$= \{(a_1 + \max(a_3 - b_1, 0), (b_3 + \max(b_1 - a_3, 0)\}$

$= \{(5+ \max(4.5 - 3.9, 0), (1.9 + \max(-0.6, 0)\}$

$= (5 + 0.6, 1.9 + 0) = (5.6, 1.9)$

As per Step 3: Using Johnson's method optimal sequence is

$S_1 = 5, 2, 4, \beta$

i.e. 5 2 4 1 3

Other optimal sequences for minimize rental cost, are

$S_2 = 2\ 5\ 4\ 1\ 3$

$S_3 = 4\ 5\ 2\ 1\ 3$

$S_4 = 1\ 3\ 5\ 2\ 4$

For S_1 = 5- 2- 4- 1- 3

The In-Out flow table for $S_1$is shown in **table 5**.

Thus, the total elapsed time = 20.3

Utilization time for M_2 = 20.3 - 2.2

= 18.1

For S_2 = 2 - 5 - 4 - 1 – 3

The In-Out flow table for $S_1$is shown in **table 6**.

Total elapsed time = 20.7

Utilization time for M_2 = 20.7 - 3.3 = 17.4

For S_3 = 4 - 5 - 2 - 1 – 3

The In-Out flow table for $S_1$is shown in **table 7**.

Total elapsed time = 22.6

Utilization time for M_2 = 22.6 - 3.4 = 19.

For S_4 =1-3-5-2-4

The In-Out flow table for $S_1$is shown in **table 8**.

Total elapsed time = 23.3

Utilization time for M_2 = 23.3-5.9 = 17.4

The total utilization of machine A is fixed 18.4 units and minimum utilization time of B machine is 17.4 for sequence S_2 and S_4.

Therefore the optimal sequence are

S_2= 2-5-4-1-3 or S_4 = 1-3-5-2-4

Total Rental Cost = 18.4 x 21 + 20.7 x 17

= 386.4 + 351.9

= 738.3 units.

References

Johnson, S.M.(1954), Optimal two & three stage production schedules with set up times includes, Nav. Res. Log. Quart. Vol 1. ,pp 61-68.

W.E.Smith (1956), Various optimizers for single stage production, Naval Research logistic 3, pp 89-66.

Ignall E and Scharge, L (1965), Application of branch and bound technique to some flow shop scheduling problems, Operation Research 13, pp 400-412.

Bagga P.C.(1969), Sequencing in a rental situation, Journal of Canadian Operation Research Society.7 ,pp 152-153.

Campbell H.A,Duder R.A and Smith M.L(1970)**,** A heuristic algorithm for the n–job, m-machine sequencing problem" Management science 16, b630-b637.

Maggu, P.L & Dass. G (1977), Equivalent jobs for job block in job sequencing, Opsearch, Vol. 14 No. 4, pp. 277-281.

Szware, W., Special cases of flow shop problem (1977) , Naval Research Logistics Quarterly, 24, pp 483-492.,

Maggu, P.L. and Das, G**.**(1980), "On idle/waiting time operator $O_{i,w}$ having applications to solution of some sequencing/scheduling problems", PAMS, Vol. XI, No., pp 1-2.

S. Szwarc (1983), The flow shop problem with mean completion time criterion, AIIE Trans. 15, pp 172-176.

Singh , T.P.(1985), On n x 2 flow shop problem solving job block, Transportation times, Arbitrary time and Break-down machine times, PAMS Vol. XXI, No. ,pp 1-2.

Adiri; I., Bruno, J., Frostig, E. and Kan; R.A.H.G (1989), Single machine flow time scheduling with a single break-down, Acta Information, Vol.26(No.7), pp 679-696.

Akturk, M.S. and Gorgulu, E (1999), Match up scheduling under a machine break-down, European journal of operational research, pp 81-99.

Chander Shekharn, K, Rajendra, Deepak Chanderi (1992), An efficient heuristic approach to the scheduling of jobs in a flow shop, European Journal of Operation Research 61, pp 318-325.

Singh, T.P., K, Rajindra & Gupta Deepak (2005), Optimal three stage production schedule the processing time and set up times associated with probabilities including job block criteria, Proceeding of National Conference FACM- (2005), pp 463-470.

Chandramouli, A.B.(2005), Heuristic approach for N job 3 machine flow shop scheduling problem involving transportation time, break-down time and weights of jobs, Mathematical and Computational Application, Vol.10 (No.2), pp 301-305.

Singh, T.P, Gupta Deepak, Minimizing rental cost in two stage flow shop , the processing time associated with probabilies including job block, Reflections de ERA, Vol 1. Issue 2, (2006), pp 107-120.

Belwal, O.K. and Mittal Amit (2008), N jobs machine flow shop scheduling problem with break-down of machines, transportation time and equivalent job block, Bulletin of Pure & Applied Sciences-Mathematics, Jan-June,2008, Source Volume 27,Source issue: 1.

Gupta Deepak, Sharma Sameer (2011), Minimizing rental cost under specified rental policy in two stage flow shop, the processing time associated with probabilities including break-down interval and Job – block criteria , European Journal of Business and Management, Vol 3, No 2, pp 85-103.

Gupta Deepak, Sharma Sameer, Gulati Naveen and payal Singla (2011), Optimal Two Stage Flow Shop Scheduling to Minimize the Rental Cost including Job Block Criteria, Set Up Times and Processing Times Associated with Probabilities ,European Journal of Business and Management, Vol 3, No 3, pp 268-286.

Note1. Idle time of *1st* machine is always zero i.e. $\sum_{i=1}^{n} I_{i1} = 0.$

Note2. Idle time of *1st* job on *2 nd* machine (I_{i2}) = Expected processing time of 1st job on machine = $A_i^{'}$.

Note3. Rental cost of machines will be minimum if idle time of *2nd* machine is minimum.

Table 1: The mathematical model of the problem

ob	**Machine A**		**Machine B**	
i	a_{i1}	p_{i1}	b_{i2}	q_{i2}
1	a_{11}	p_{11}	b_{12}	q_{12}
2	a_{21}	p_{21}	b_{22}	q_{22}
3	a_{31}	p_{31}	b_{32}	q_{32}
4	a_{41}	p_{41}	b_{42}	q_{42}
-	-	-	-	-
n	a_{n1}	p_{n1}	b_{n2}	q_{n2}

Table 3: The Expected processing time is

Jobs	$\mathbf{A'_i}$	$\mathbf{B'_i}$
1	5	3.9
2	3.3	4.2
3	4.5	1.9
4	3.4	2.3
5	2.2	5.1

Table 4: The expected processing time for the equivalent job on fictitious machine is

Jobs	$\mathbf{A'_i}$	$\mathbf{B'_i}$
β	5.6	1.9
2	3.3	4.2
4	3.4	2.3
5	2.2	5.1

Table 5: The In-Out flow table for S_1

Jobs	A		B	
	In	Out	In	Out
5	0	2.2	2.2	7.3
2	2.2	5.5	7.3	11.5
4	5.5	8.9	11.5	13.8
1	8.9	13.9	13.9	17.8
3	13.9	18.4	18.4	20.3

Table 6: The In-Out flow table for S_2

Jobs	A		B	
	In	Out	In	Out
2	0	3.3	3.3	7.5
5	3.3	5.5	7.5	12.6
4	5.5	8.9	12.6	14.9
1	8.9	13.9	14.9	18.8
3	13.9	18.4	18.8	20.7

Table7: The In-Out flow table for S_3

Jobs	A		B	
	In	Out	In	Out
4	0	3.4	3.4	5.7
5	3.4	5.6	5.7	10.8
2	5.6	8.9	10.8	15
1	8.9	13.9	15	18.4
3	13.9	18.4	18.4	20.3

	3.9	8.4	8.4	2.6

Table 8: The In-Out flow table for S_4

Jobs	**A**		**B**	
	In	**Out**	**In**	**Out**
1	0	5.0	5.0	8.9
3	5.0	9.5	9.5	11.4
5	9.5	11.7	11.7	16.8
2	11.7	15	16.8	21
4	15	18.4	21	23.3