IOSR Journal of Mathematics (IOSR-JM)
e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 12, Issue 6 Ver. IV (Nov. - Dec.2016), PP 82-88
www.iosrjournals.org

Solving Earliness and Tardiness Single Machine Scheduling Problem with Common Due Date

Abdelaziz Hamed[1] and Elssayed Abkar[2]
[1]*Department of Mathematics, Faculty of Science, University of Baha, King of Saudi Arabian.*
[2] *Department of Mathematics, Faculty of Science, University of Bahri , Sudan Bahri,*

Abstract: *This paper present a single machine scheduling problem with common due date to minimize the earliness and tardiness (E/T). Earliness and tardiness are considered harmful to profitability. Earliness causes inventory carrying costs and possible loss of product quality, while tardiness causes loss of customer goodwill and delay of payment. Thus the scheduling problem of minimizing the total sum of earliness and tardiness with a common due date on a single machine is important task in scheduling problem and delivery of goods in production plants, and it is known to be NP- Hard problem. In this paper a heuristic algorithm is presented and can be performed manually for small systems and by using Java language for large systems*
Keyword: *single machine scheduling, common due date, heuristic algorithms, earliness and tardiness.*

I. Introduction

In this paper we consider the static scheduling. In other words, the set of jobs to be scheduled is known in advance and is simultaneously available. Consider the problem of sequencing n independent jobs on a single machine with common due date d. For job *i,* the completion time, the earliness and tardiness, will be denoted by C_i, E_i and T_i respectively. Let *S* denote an arbitrary schedule. Then,

$$E_i = \max(0, d - C_i) \qquad T_i = \max(0, C_i - d).$$

Using the notations and definitions of Panwalker et al. (1982) we can formulate the cost function for the due date assignment problem for single machine as

$$f(d,S) = \sum_{i=1}^{n} (P_1 d + P_2 E_i + P_3 T_i). \qquad (1)$$

In the cost function equation (1), P_1, P_2, and P_3, respectively, denote the per unit due date, earliness, and tardiness costs. For the common due date assignment and scheduling the objective is to select d and to schedule the jobs such that *f*(d,S) is minimized. To describe a E/T model for the single machine from equation (1), let $P_1 = 0$, then equation (1) becomes,

Then the problem reduces to a pure scheduling problem for single machine. Actually equation (2) describes earliness and tardiness problem when the penalties of earliness and tardiness are equal. In the case of non-equal

$$f(S) = \sum_{i=1}^{n} (P_2 E_i + P_3\ T_i). \qquad (2)$$

penalties, the cost function of equation (2) for single machine case becomes

$$f(S) = \sum_{i=1}^{n} (\alpha_i E_i + \beta_i\ T_i) \qquad (3).$$

In the cost function of equation (3) α_i and β_i denote per unit earliness, and tardiness costs for J_i , respectively.

3 …n. it is assumed that all jobs are ready for processing at time zero and have the same common due date the problem is to find the order of jobs in which these *n* jobs should be processed so as to minimize the sum of total earliness and tardiness costs. The common due date is assumed to be less than the total processing time.

II. Literature Review

Job scheduling or sequencing has a wide variety of application, from designing the product flow chart and order in a manufacturing facility to modeling queues in service industries. The problem of earliness and tardiness single machine with common due date has been consider by baker and scudder (1990). All the jobs considered have a common due date. The objective was to find optimal schedules which minimize the total earliness and tardiness. Since then, the problems have been studied under different environments. Later Nordin Haji Mohamad and Fatimah Said (2011) Studied Single Scheduling with common due date. They provided an algorithm to solve the problem.

III. A Heuristic Algorithm.

Below, we present Nordin Haji Mohamed and Fatimah a heuristic algorithm for solving n-jobs single machine scheduling problem with common due date. In this paper we need to use his algorithm and writing Java language to implement for solving small and large systems.

Given n jobs to be processed on a single machine, the processing time of jobs i being t_i i= 1, 2, 3… n. It is assumed that all jobs are ready for processing at time zero and have the same common due date. The problem is to find the order or schedule in which these n jobs should be processed so as to minimize the sum of total earliness and tardiness costs. The common due date, is assumed to be less than the total processing time.

Step 0: Sort and number the n jobs in non increasing order of processing time t_i *(i= 1,2,……. n)* Such that $t_1 \geq t_2 \geq t_3 \ldots\ldots \geq t_i \geq t_{i-1} \ldots\ldots t_n$. In genera,l we can represent the jobs in tabular form,

J_i	J_1 j_2 ….. …… j_{i-1} j_i j_{i+1} …… j_n
t_i	t_1 t_2 ….. …… t_{i-1} t_i t_{i+1} …… t_n

Compute

Total processing time, $T = \sum_{i=1}^{n} t_i$

$T_0 = T - D$ and $E_0 = D$

Introduce two empty sets, $S_0^E = \emptyset$ and $S_0^T = \emptyset$.

Step 1: Consider job j_1 with processing time t_i = max (t_i,*1,2,3,……n*)

Set $T_1 = T_0$ and $E_1 = E_0$

If $T_1 < E_1$ set $S_1^E = S_0^E = \{J_1\}$ and $S_1^T = S_0^T$

If $T_1 \geq E_1$ set $S_1^T = S_0^T + \{J_1\}$ and $S_1^E = S_0^E$

Step *i* : consider job J_i with processing time t_i ,*(1< i≤ n)*

If previous job, $J_{i-1} \in S_{i-1}^E$ compute $T_i = T_{i-1}$ and $E_i = E_{i-1} - t_{i-1}$

If previous job, $J_{i-1} \in S_{i-1}^T$ compute $T_i = T_{i-1} - t_{i-1}$ and $E_i = E_{i-1}$

Assignment decision

If $T_i < E_i$ set $S_i^E = S_{i-1}^E + \{J_i\}$ and $S_i^T = S_{i-1}^T$

If $T_i \geq E_i$ set $S_i^T = S_{i-1}^T + \{J_i\}$ and $S_i^E = S_{i-1}^E$

Iteration terminates when all jobs have been assigned to either S_n^E or S_n^T

Scheduling decision

Check (and sort if necessary) so that:

S_n^E = {jobs in non-increasing order of processing times},

S_n^T = {jobs in non-decreasing order of processing times}

The optimal schedule S*= { S_n^E,S_n^T}in other words , jobs are schedule according to their sequence in S_n^E followed by S_n^T

Note that the procedure only involves **n** enumerations (iterations) as compared to **n!** possible schedules.

We illustrate the above algorithm by considering the example below:

jobs		J_1	J_2	J_3	J_4
p_i		8	10	4	7
	α	1	1	1	1
	β	1	1	1	1

According to jobs scheduling which was done by abdelaziz Hamed and Bahrom Sanugi (2002) using neural network approach for above dataTheir schedule result showed that j_2,j_3,j_4, j_1 with total cost equal 26 units.
So we want presented this algorithm for above example
Step 0: Sort and number the 4th jobs in non-increasing order of processing time t_i

The table below show that:

jobs	J_2	J_1	J_4	J_3
p_i	10	8	7	4

Step1:- $T = \sum_{i=1}^{n} t_i = 29$ and D = 15
Introduce two empty set $S_0^E = \emptyset$ and $S_0^T = \emptyset$
T_0=29-15 =14 then and E_0= 15 Set $T_1 = T_0$and E1= E_0
t_1= 10
$T_1 < E_1$ set $S_1^E = S_0^E + \{J_1\}$ and$S_1^T = S_0^T$ so $J_1 \in \{ S_n^E \}$

Step 2: consider J_2 with processing time t_2= 8
Previous job$J_1 \in S_n^E$Compute $T_2 = T_1$ and $E_2 = E_1 - t_1$
T_2 = 14 and E_2 = 15-10 = 5
Assignment decision is:
$T_2 > E_2$ set $S_2^T = S_1^T + \{J_2\}$ and$S_2^E = S_1^E$ so $J_2 \in \{ S_n^T \}$

Step 3: consider J_3 with processing time t_3= 7
Previous job$J_2 \in S_n^T$Compute $T_3 = T_2 - t_2$and $E_3 = E_2$
T_3 = 14-8 =6 and E_3 = 5
Assignment decision is:
$T_3 > E_3$ set $S_3^T = S_2^T + \{J_3\}$ and $S_3^E = S_2^E$ so $J_3, J_2 \in \{ S_n^T \}$

Step 4: consider J_4 with processing time t_4= 4
Compute $T_4 = T_3 - t_3$and $E_4 = E_3$
T_4 = 6-7 =-1 and E_3 = 5
Assignment decision is:
$T_4 < E_4$ set $S_4^E = S_3^E + \{J_4\}$and $S_4^T = S_3^T$ so $J_4, J_1 \in \{ S_n^E \}$
S_n^E= {jobs in non-increasing order of processing times}
S_n^T= {jobs in non-decreasing order of processing times}
The optimal schedule S*= { S_n^E,S_n^T}in other words , jobs are schedule according to their sequence in S_n^E followed by S_n^T .
End of iteration since all jobs have been assigned to either S_4^E , or S_4^T
Then we observe that jobs in S_4^E= {J1,J_4} are in non-increasing order but in S_4^T= { J_2 , J_3 } are should be in non decreasing order so
S_4^* = { S_4^E , S_4^T } as shown { J_1, J_4, J_2, J_3 } according to our sorting in first time $J_1 = \underline{J_2}$, $J_4 = \underline{J_3}$, $J_2 = \underline{J_1}$, $J_3 = \underline{J_4}$ so our scheduling will be as
J_2 , J_3 , J_1 , J_4. And the cost is equal 26 units.

below: S_4^* = { S_4^E , S_4^T }

jobs	J_2	J_3	J_1	J_4
p_i	10	4	7	8
C_i	10	14	21	29
I C_i-D I	5	1	6	14

Giving F1= $\sum_{i=1}^{4} |C_i - D|$ = 26

We compare between the two algorithms we get the same result of scheduling with cost 26 units

IV. Java Application Language Program (JALP)

So in this Section we designed computer program by (JALP) to solve the problem of scheduling for large number of jobs or system i.e. 100 jobs to be scheduled in one machine which it will became very complicated although its solvable when we use our previous heuristic algorithm, but (JALP) was easy and its accurate as well as our heuristic algorithm, so If we want to applied our previous example with Java application language program the result is to be optimal and show that

J_2 , J_3 , J_1 , $J_4$with the cost is equal 26 units. For the Software and the result see appendix I and II

V. Conclusion

In this paper we have presented Nordin Haji Mohamed and Fatimah a heuristic algorithm for solving n-jobs single machine scheduling problem with common due date. And we used it with Java language to implement for solving small and large systems. We illustrate with two examples one with manual approach and another with Java software we get an optimal solution in two examples. Future research can be carried out on studying similar models in multi machine environment.

References

[1] Abdelaziz Hamed and Bahrom Sanugi (2011) . "Neural Network and Scheduling" Germany : lap Lambert Academic publishing
[2] Abdelaziz Hamed (2002) . Application of neural network for solving Job scheduling problems. PhD thesis. UTM, Malaysia.
[3] Baker, K. and Scudder, G. (1989). "On the Assignment of Optimal Due Date." Journal of Operational Research Society. 40; 93-95.
[4] Baker, K. and Scudder, G. (1990). "Sequencing with Earliness and Tardiness Penalties: A review." Operations Research. 38; 22 - 36.
[5] Nordin Haji and Fatimah Said (2011)). " Solving Single machine Scheduling problem with common due date. " Business Management Dynamics. 4; 63-72.

Appendix I
Java language Software application:

```
public static void main(String[] args) {
    Scanner key = new Scanner(System.in);
    int T[] = new int[100];
    int E[] = new int[100];
    int SE[] = new int[100];
    int ST[] = new int[100];
    int J[] = new int[100];
    int S[] = new int[100];
    int t[] = new int[100];
    int due_date, Time;
    System.out.print("Enter the Due date = ");
    due_date = key.nextInt();      //insert the due date
    int n;           //define the number of jobs
    System.out.print("Enter the Number of jobs =");
```

```
n = key.nextInt();
for (int i = 1; i <= n; i++) {
    System.out.print("Enter the process time  t[" + i + "]=");
    t[i] = key.nextInt();
}
int sum = 0;
System.out.print("\n ********** The Process Time Befor Sorting ************\n\n  t[i] = ");
for (int i = 1; i <= n; i++) {
    System.out.print("[ " + t[i] + " ]");
    sum = sum + t[i];
}
Time = sum;
System.out.print("\n\n ************ The Summation Of Process Time ************\n\n\t\t T =" + Time)
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        if (t[j] < t[j + 1]) {
            int temp = t[j + 1];
            t[j + 1] = t[j];
            t[j] = temp;
```

```
System.out.print("\n\n *********** The Procss Time After Sorting ************\n\n t[i] =");
for (int i = 1; i <= n; i++) {
    J[i] = t[i];
    System.out.print("[ " + t[i] + " ]");
}
T[0] = Time - due_date;
E[0] = due_date;
T[1] = T[0];
E[1] = E[0];
int i = 1;
int e = 1, s = 1;
while (i <= n) {
    if (T[i] > E[i]) {
        ST[e] = J[i];
i]=SE[i-1];
        T[i + 1] = T[i] - t[i];
        E[i + 1] = E[i];
        e++;
    } else {
```

```
                    SE[s] = J[i];
'/ST[i]=ST[i-1];
                    T[i + 1] = T[i];
                    E[i + 1] = E[i] - t[i];
                    s++;
                }
                i++;
            }
            for (int v = 1; v < e; v++) {
                System.out.print("\nST [" + v + " ] = " + ST[v] + "\n");
            }
            for (int u = 1; u < s; u++) {
                System.out.print("\nSE [" + u + " ] = " + SE[u] + "\n");
            }
            int z = e + s;
            for (i = 1; i < z; i++) {
                if (i < s) {
                    S[i] = SE[i];
                } else {
                    S[i] = ST[--e];
            }
            int C[] = new int[100];
            int DE[] = new int[100];
            int DEA[] = new int[100];
            C[0] = 0;
            int CA[] = new int[100];
            CA[0] = 0;
            for (i = 1; i < z; i++) {
                C[i] = C[i - 1] + S[i];
                DE[i] = Math.abs(C[i] - due_date);
                CA[i] = CA[i - 1] + t[i];
                DEA[i] = Math.abs(CA[i] - due_date);
```

```
    System.out.println("\t\t ---------- The Jobs Befor Scheduling---------- \n");
    System.out.print("\n J[ i ]      = ");
    for (i = 1; i < z - 1; i++) {
        System.out.print("J[" + i + "]" + "\t");
    }
System.out.print("\n ----------
System.out.print("\n t[ i ]        = ");
for (i = 1; i < z - 1; i++) {
    System.out.print(t[i] + "\t");
}
System.out.print("\n ----------
  System.out.print("\n ----------
  System.out.print("\n C[ i ]       = ");
  for (i = 1; i < z - 1; i++) {
      System.out.print(CA[i] + "\t");
  }
  System.out.print("\n ----------
  System.out.print("\n |C[ i ] - D| = ");
  for (i = 1; i < z - 1; i++) {
      System.out.print(DEA[i] + "\t");
  }
```

```
            System.out.print("\n ----------------
            int ff = 0;
            for (i = 1; i < z - 1; i++) {
                ff = ff + DEA[i];
            }
            System.out.print("\n ----------------
            int ff = 0;
            for (i = 1; i < z - 1; i++) {
                ff = ff + DEA[i];
            }
        System.out.println("\n\t\tF* = " + ff + " (penalty days)");
        System.out.println("\n\n\t\t ---------- The Jobs After Scheduling---------- \n");
        System.out.print("\n S[ * ]      = ");
        for (i = 1; i < z - 1; i++) {
            System.out.print(S[i] + "\t");
        }
    System.out.print("\n --------------------------------
    System.out.print("\n C[ i ]        = ");
    for (i = 1; i < z - 1; i++) {
        System.out.print(C[i] + "\t");
    }
System.out.print("\n ------------------------------------------------
System.out.print("\n |C[ i ] - D| = ");
for (i = 1; i < z - 1; i++) {
    System.out.print(DE[i] + "\t");
}
    System.out.print("\n ------------------------------------------------
    int f = 0;
    for (i = 1; i < z - 1; i++) {
        f = f + DE[i];
    }
    System.out.println("\n\t\t\t F* = " + f + " (penalty days)");
```

Appendix II

The Applications of the software above for our example and its result 1/

```
run:
Enter the Due date = 15
Enter the Number of jobs =4
Enter the process time  t[1]=8
Enter the process time  t[2]=10
Enter the process time  t[3]=4
Enter the process time  t[4]=7
```

```
********** The Process Time Befor Sorting *************

 t[i] = [ 8 ][ 10 ][ 4 ][ 7 ]

************ The Summation Of Process Time ************

                T =29

************ The Procss Time After Sorting *************

 t[i] =[ 10 ][ 8 ][ 7 ][ 4 ]
ST [1 ] = 8

ST [2 ] = 7

SE [1 ] = 10

SE [2 ] = 4
  ---------------- The Jobs Befor Scheduling--------------------
```

J[i]	= J[1]	J[2]	J[3]	J[4]	
t[i]	= 10	8	7	4	Jobs Title
C[i]	= 10	18	25	29	Jobs Time
|C[i] - D|	= 5	3	10	14	Completion time
					Comleted Time After the desired

F* = 32 (penalty days)

---------- The Jobs After Scheduling----------

S[*]	= 10	4	7	8	
C[i]	= 10	14	21	29	Time of Jobs Scheduling
|C[i] - D|	= 5	1	6	14	Completion Time
					Comleted Time After the desired

F* = 26 (penalty days)

BUILD SUCCESSFUL (total time: 1 minute 17 seconds)