Indian Journal of Science and Technology, *Vol 9(37), DOI: 10.17485/ijst/2016/v9i37/97527, October 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Single Machine Scheduling Model with Total Tardiness Problem

Neelam Tyagi[1*], R. P. Tripathi[1] and A. B. Chandramouli[2]

[1]Department of Mathematics, Graphic Era University, Dehradun - 248002, Uttarakhand, India; neelam24tyagi@gmail.com, tripathi_rp0231@rediffmail.com
[2]Department of Mathematics, Meerut College, Meerut - 250001, Uttar Pradesh, India; dr.abchandramouli@gmail.com

Abstract

Objectives: In this paper, Five Dispatching Rules and a Branch & Bound algorithm is introduced for Single Machine Total Tardiness Scheduling Problem (SMTTSP) to minimize the total (average) tardiness and number of tardy jobs. **Methods/ Statistical Analysis:** We proposed five dispatching (priority) rules as Shortest Processing Time, Earliest Due Dates, Longest Processing Time, Minimum Slack Time and First Come First Serve for SMTTSP and compared the performance of all the proposed priority rules. Furthermore, a numerical illustrations is also provided to select the best dispatched rule of SMTTSP. Next, we developed a Branch & Bound Algorithm for SMTTSP using best selected dispatching rule. We also developed Branch Tree to understand the Lower bound process of the B& B Algorithm. **Findings:** The main aim to proposed these dispatching rules and a Branch and Bound Algorithm is to obtain the optimal sequence to optimize the total (average) tardiness and number of total tardy jobs. The comparative analysis of the dispatching rules shows that EDD rule is better than other dispatching rules for minimization of total tardy jobs and tardiness of the jobs while SPT rule is better for minimization of make span. But Dispatching rules do not have the guarantee to give an optimal solution. So in this study, an exact algorithm (Branch & Bound) was developed with EDD rule for finding the optimal solution for SMTTSP. The comparative study between dispatching rules and an exact (B&B) algorithm is being justified by numerical illustrations and we found that the EDD rule and B&B Algorithm give the same results. Hence it was concluded that EDD rule works as an Exact algorithm and gives the optimal solution for SMTTSP. **Application/Improvements:** The computational results of the proposed (B&B) algorithm and Dispatching rules show that our methodology is more useful than other optimal approach for SMTTSP and it provides an important tool for decision makers.

Keywords: Average Completion Time, (Average) Tardiness of Jobs, Branch and Bound Algorithm, Branch Tree, No of Tardy Jobs, Single Machine Scheduling

1. Introduction

The single machine scheduling problem with Total Tardiness (TT) is the biggest research difficulty of scheduling problems in the real world. In recent years, Just-In-Time (JIT) scheduling was applied in a lot of studies to solve the problems which consist of tardiness and earliness of jobs. In JIT scheduling most of the production industries are referred to meet the clients requirement in terms of due dates as possible for avoiding the earliness and tardiness penalties[1-3]. When job is delivered later to its due date is called tardy job. Furthermore, if the job is delivered earlier to its due date is called early jobs[4]. The early and tardy jobs are penalized to a great extent. Consequently, we called the idealistic scheduling as all the jobs are finished incisively on its due dates. This paper focuses on Single Machine Total Tardiness Scheduling Problem (SMTTSP) with zero ready times, $r_j = 0$. For single machines there is only one resource or machine available for processing of jobs or tasks. In the single machines scheduling, n jobs $(j_1 1, j_2, j_3, j_1 4, \ldots \ldots j \square_1 n)$ set are processed on a single machine $\mathbf{M}$ with due dates D_i and processing times $p\_i\ (where\ i = 1\ to\ n)$. Due

**Author for correspondence*

date is the important constraint of single machine scheduling. The Lateness $L_{ti} = \{(C_i - D_i)\}$ either positive or negative tardiness. The positive deviation between completion time (C_i) and due dates (D_i) is called the tardiness, $T_{di} = \{\max[(C_i - D_i, 0)]\}$ and also called positive part of lateness. Besides negative Part of the lateness is called Earliness, $E_i = \{\max[(0, D_i - C_i)]\}$. In [5] the classification our problem is characterized as $1\|D_i(i)\|\sum T_i(di)$. There are two phases of the paper in first phase we proposed five dispatching rules (SPT, EDD, LPT, FCFS and MST) and select the best dispatch rule for minimization of total tardy jobs and total tardiness or average tardiness of all the jobs. In the second phase of the paper we have used best selected dispatch rule from the first phase and developed an exact algorithm as Branch and Bound (B & B) algorithm for minimization of total tardiness. The performances of all dispatching rules and Branch & Bound algorithm are justified by numerical illustration.

The remaining paper is formulated as follows. In next section 2 we review the scheduling literature related to numerous versions of total tardiness for single machine. Section 3 present the brief description of this scheduling problem. Section 4 is devoted to classify the dispatching (Priorty) rules and in the section 5, we proposed Dispatching rules for tardiness problem. In section 6, we solved the numerical with the help of dispatching rules and study the comparison between them. In Section 7, we developed Branch & Bound algorithm and study the lower bound. In section 8 we solve the numerical with the help of B&B algorithm and calculate the lower bound using objective function. In this section we also present the branch tree to represent the lower bound process. Finally, the concluded remarks and future research are drawn in the Section 9.

In 1961, Total Tardiness (TT) scheduling problem was done firstly. In the past two decennium, various dispatching or priory rules are developed. In[6] studied EDD rule in 1954. In[7] have been proposed a comprehensive research survey on heuristic dispatching rules. In[8] proposed novel scheduling rules that synthesized distinct dispatching rules based on objective function of due dates. In[9] presented the work in which non preemptive sequence is obtained from any preemptive sequence with least tardiness. firstly [10] was precisely innovate the due date based scheduling problems[11] and[12] differentiated the dispatching (priorty) rules. In[3] reviewed the latest theoretical developments for SMTTP and in his paper he also reviewed about approximation and exact algorithms for $1\|\sum T_i(di)$ problem. The minimization of total tardiness scheduling problems demonstrated as NP-hard Due to complexity of SMTTSP by[13] that means it is impossible to find an optimal solution without using the enumerative algorithm. In[14] proposed an algorithm to optimize the tardy jobs in SMTTSP. In[15] derived the necessary conditions to evaluate the p_i and d_i to obtain the optimal sequence using EDD or SPT rulel. In exact algorithm most effective algorithm is the Branch & Bound algorithm that gives the guarantee of optimality. The method was first proposed by[16] for discrete programming. In[17] proposed Branch & Bound algorithm for enumerative methods. In[18] developed an effective B&B algorithm to solved single machine scheduling problem capable to 100 jobs. A Branch and Bound algorithm was developed by [19] to minimize the total tardiness. In[20] developed a quick exact algorithm(B & B) to optimize the Total Tardiness (TT) without using the Lower Bound in Branch & Bound algorithm. In[21] developed a novel B & B algorithm for SMTTSP and in[22] remonstrated to their points regarding the Branch and Bound algorithm of[21]. In[23] remarked that excision of the LB (Lower Bound) meliorate the functioning of their Branch and Bound (exact) algorithm. In[24,25] proposed an effective algorithm to optimize the tardiness of the jobs in single machine scheduling problem using Branch and Bound algorithm.

In[26] presented an extensive survey of T/E problems in the single machine scheduling included discourse of the unrestricted mutual due dates. In[27] studied the SMTTP for minimization of the TT (Total Tardiness) of n equal length preemptive jobs to the single machine scheduling. In[28] developed an algorithm for SMTTP in polynomial time within the presence of deadlines. In[29] presented an algorithm for SMTTP with release dates and preemption jobs. Recently[30] survey the modish theoretical improvements for the SMTTP.

2. Problem Description

In this paper we consider the single machine total tardiness scheduling problem with due dates $(1\|D_i\|\sum T_{di})$. where "n" jobs have be processed on the single machine M. The goal of the paper is to obtain the optimal sequence to minimize the total (average) tardiness and number of tardy jobs. We used Dispatching Rules and Exact Algorithms for minimization of tardiness of jobs. The structure of our SMTTSP is presented in Figure 1.

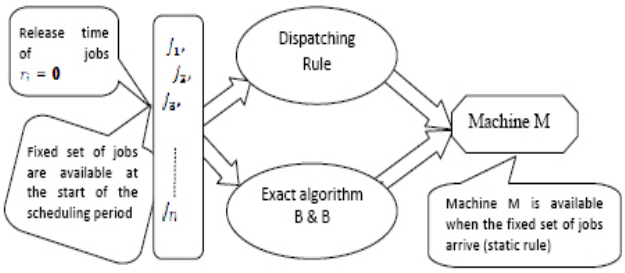

Figure 1. Structure of single machine scheduling problem.

2.1 Assumptions

i. The entire jobs and the machine are functioning at Zero time.
ii. The Problem having the set of fixed jobs which are to be all completed. The number of jobs doesn't change. (Static Scheduling problem)
iii. The jobs related information is known beforehand. This information includes processing time (p_i), due dates (D_i) and release time of jobs $r_j = 0$. (Deterministic scheduling problem)
iv. Only one operation will be operated by a single machine at one mentioned time.
v. Once machine is started to execute the operation it should be completed first, only after that the next operation will be execute. Hence, preemption is not allowed.
vi. The machine breakdown is not occurring and it is presumed to operate endlessly.
vii. Setup times are included in the processing time.

2.2 Notations and Parameters used

i. M = Machine
ii. p_i = Processing times of i jobs. Where $(i = 1 \text{ to } n)$
iii. m_i = Processing times of i jobs on M Machine
iv. D_i = Due Dates of i job
v. C_i = Completion Time of i job
vi. $\overline{C_i}$ = Mean Completion Time
vii. $\sum_{i=1}^{n} C_i$ = **Total Completion time of** □□□□□ or Makespan.
viii. T_{di} = Tardiness of i job
ix. $\sum_{i=1}^{n} T_{di}$ = **Total Tardiness of** □□□□□
x. $\overline{T_{di}}$ = Mean Tardiness
xi. N_T = No of Tardy Jobs
xii. $A(n)$ = Set of sequencing "n" jobs
xiii. $LB(n)$ = Lower Bound of "n" job
xiv. j_u = unscheduled job
xv. j_s = scheduled job
xvi. T_{ju} = Tardiness of Unscheduled job
xvii. $T_{j_{ud}}$ = Tardiness of unscheduled job with common due date
xviii. T_{js} = Tardiness of Scheduled job
xix. C_{ju} = Completion time of unscheduled job
xx. d = Common Due date

2.3 Objective Function

1. $\min \left\{ \sum_{i=0}^{n} T_i \right\}$ = **Total Tardiness**
2. $\min\{T_{dmax}\}$ = **maximum tardiness**
3. $\min \left\{ \sum_{i=0}^{n} N_T \right\}$ = **Number of Tardy(late)Jobs**
4. $\min\{(T_i di)^- \}$ = **Average Tardiness**

2.4 Mathematical Model for Single Machine Tardiness Problem

Consider a set of n jobs $(j_1, j_2, j_3, j_4, \square \ldots\ldots j_n)$ with processing time $(p_1, p_2, p_3, p_4, p_5 \ldots\ldots p_n)$ and due dates $(d_1, d_2, d_3, d_4, d_5, \ldots\ldots d_n)$ respectively are processed on single machine **M**. This model in a matrix form is represented in Table 1.

Table 1. mathematical model of single machine scheduling with due dates

Jobs (i)	Machine M Processing time	Due Dates (D_i)
j_1	p_1	d_1
j_2	p_2	d_2
j_3	p_3	d_3
⋮	⋮	⋮
j_n	p_n	d_n

3. Dispatching Rules for Tardiness Problems

Dispatching rules are the most classic and well-known methods to build a schedule. Dispatching rules are a very common means of scheduling due to their simplicity, speed, and predictability of speed in arriving at a solution. It is also called priority rules, sequencing rules, scheduling rules decision rules. Dispatching rules are defined according to job parameters, machine parameters and shop characteristics. When the priority of each job is determined, jobs are sorted and then the job with the highest priority is selected to be processed first. In some cases dispatching rules can give exhaustive optimum, but most of the problems they are heuristic. Dispatching rules are also often implemented without an expert system. The biggest drawback of many dispatching rules is the quality of the solution. There is no guarantee of dispatching rules to give an optimum solution.

3.1 Classification of Dispatching Rule

In[11] and[12] shown the classification of dispatching rules are as follow:

- **Local Rules:** these rules are connected with the local operable data.
- **Global Rules:** Global rules are applied to discharge the jobs with the help of data's information exists on the shop floor.
- **Static Rules:** Static Rules are not changed or vary throughout time. These rules neglect the position of the jobs in the shop floor. "We used the static rule in this problem"
- **Dynamic Rules:** These rules are time dependent. Dynamic rules are changed consequently to the position of the jobs in the shop floor.
- **Forecast Rules:** Forecast rules are applied to provide the priority of the jobs.

We show the Classification of Dispatching Rule in Figure 2.

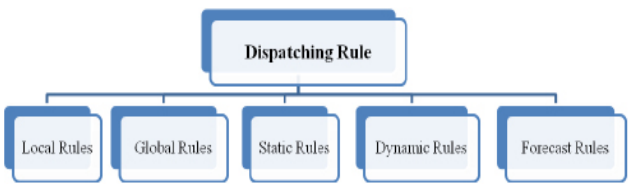

Figure 2. Classification of Dispatching Rule.

4. Proposed Dispatching Rules for Single Machine Tardiness Problem

4.1 SPT (Shortest Processing Time) Rule

It is also called Shortest expected processing time if processing times are connected with their respective probabilities. The Jobs are arranged or scheduled accordingly non decreasing order of their processing time; the job having least processing time is operated initially. If there are n jobs $\{j_1, j_2, j_3, j_4, j_5 \dots \dots \dots j_n\}$ with processing time $[\{p]_1, p_2, p_3, p_4, p_5, \dots \dots \dots \dots p_n\}$ and due dates $D_i = [\{d]_1, d_2, d_3, d_4, d_5, \dots \dots \dots \dots d_n\}$ respectively are processed on single machine **M**, then, according to SPT rule, sequenced the jobs in non- decreasing order respective to processing time $p_1 \leq p_2 \leq p_3 \leq p_4 \leq p_5 \dots \dots \dots p_n$ to minimize the objective function.

4.2 EDD (Earlier Due Date) Rule

According to this rule, the job having least due date is operated first. The jobs are arranged in raising order of their respective due dates; the job having earliest due date will be processed first similarly the job having second earliest due date will be processed second and this process will be continue until all the jobs are sequenced.. Hence, the jobs are sequenced according to increasing order of their due dates $(d_i j)$

$$d_1 \leq d_2 \leq d_3 \leq d_4 \leq d_5 \leq \dots \dots \dots d_n$$

R. Jackson was the first who studied EDD in 1954. Hence, we can also called dispatching EDD rule to Jackson's rule.

4.3 LPT (Longest Processing Time)

It is also called Longest Expected processing Time (LEPT). According to this rule the job having largest processing time will be sequenced first with the largest processing time is processed first and this process will be continue until all the jobs are sequenced.

$$p_1 \geq p_2 \geq p_3 \geq p_4 \geq p_5 \geq \dots \dots \dots p_n$$

4.4 MST (Minimum Slack Time) Rule

It is a variant of EDD rule. It subtracts "remaining total shop time for the job" including that of the operation

being scheduled from the "time remaining until the due date". The resulting value is called "slack". Slack time =(due date - today's date) - (remaining processing time). Jobs are run in the order of the smallest amount of slack.

$$\left\{(d]_1 - p_1) \leq (d]_2 - p_2) \leq (d]_3 - p_3) \leq (d]_4 - p_4) \leq \square \ldots (d]_i - p_i) \leq (d]_{i+1} - p_{i+1}) \leq \ldots\ldots (d]_n - p_n)\right\}$$

4.5 FCFS (First Come First Serve) Rule

The job which arrives the work station earlier than the others has the highest priority and is to be scheduled next. According to this rule, the job which come first inside the shop floor will be sequenced at the first position. İn the similar way the job which will entered second in the shop floor will be sequenced second and the same process will be continued untill all the jobs are sequenced. We can see this type of sequence in the bank queuing system.

5. Numerical Illustrations

Let 4 jobs are being processed on single machines (M) with their processing time (p_i) associated with due dates $(D]_i)$ are given in Table 2. Numerical solved by all proposed dispatching rules are as in Table 3 and Comparative result between dispatching rules is shown in Table 4.

Table 2. Four jobs single machine scheduling problem with due dates

Jobs (j_i)	Processing time (m_{i1})	Due dates (D_i)
1	8	12
2	5	16
3	1	4
4	10	18

Table 3. Four jobs single machine scheduling problem with due dates solved by Dispatching Rules

SPT RULE				
Jobs (j_i)	Processing time (m_{i1})	Due dates (D_i)	Completion time (C_i)	Tardiness $T_{di} = \{\max(C]_i - D_i, 0)\}$
3	1	4	1	0
2	5	16	6	0
1	8	12	14	2
4	10	18	24	6
$N_T = 2, \quad \overline{C_i} = 45 \div 4 = 11.25, \quad \overline{T_{di}} = 8 \div 4 = 2,$			$\sum_{i=1}^{4}(C_i) = 45$	$\sum_{i=1}^{4}(T_{di}) = 8$
EDD RULE				
Jobs (j_i)	Processing time (m_{i1})	Due dates (D_i)	Completion time (C_i)	Tardiness $T_i = \{\max(C]_i - D_i, 0)\}$
3	1	4	1	0
1	8	12	9	0
2	5	16	14	0
4	10	18	24	6
$N_T = 1, \quad \overline{C_i} = 48 \div 4 = 12, \quad \overline{T_{di}} = 6 \div 4 = 1.5,$			$\sum_{i=1}^{4}(C_i) = 48$	$\sum_{i=1}^{4}(T_{di}) = 6$
FCFS RULE				
Jobs (j_i)	Processing time (m_{i1})	Due dates (D_i)	Completion time (C_i)	Tardiness $T_{di} = \{\max(C]_i - D_i, 0)\}$
1	8	12	8	0

(*Continued*)

2	5	16	13	0
3	1	4	14	10
4	10	18	24	6
$N_T = 2, \quad \overline{C_i} = 59 \div 4 = 14.78, \quad \overline{T_{di}} = 16 \div 4 = 4,$			$\sum_{i=1}^{4}(C_i) = 59$	$\sum_{i=1}^{4}(T_{di}) = 16$

LPT RULE

Jobs (j_i)	Processing time (m_{i1})	Due dates (D_i)	Completion time (C_i)	Tardiness $T_{di} = \{\max[(C]_i - D_i, 0)\}$
4	10	18	10	8
1	8	12	18	6
2	5	16	23	7
3	1	4	24	20
$N_T = 4, \quad \overline{C_i} = 75 \div 4 = 18.75, \quad \overline{T_{di}} = 41 \div 4 = 10.25,$			$\sum_{i=1}^{4}(C_i) = 75$	$\sum_{i=1}^{4}(T_{di}) = 41$

MST RULE

Jobs (j_i)	Processing time (m_{i1})	Due dates (D_i)	Completion time (C_i)	MST $[(D]_i - m_{i1})$	Tardiness $T_i = \{\max[(C]_i - D_i, 0)\}$
3	1	4	1	3	0
1	8	12	9	4	0
4	10	18	19	8	1
2	5	16	24	11	8
$N_T = 2, \quad \overline{C_i} = 53 \div 4 = 13.25, \quad \overline{T_{di}} = 41 \div 4 = 10.25,$			$\sum_{i=1}^{4}(C_i) = 53$		$\sum_{i=1}^{4}(T_{di}) = 9$

Table 4. Comparative results of Dispatching Rules

	SPT RULE	EDD RULE	FCFS RULE	LPF RULE	MST RULE
$\sum_{i=0}^{4} T_{di}$	8	6	16	41	9
$\overline{T_{di}}$	2	1.5	4	10.25	10.25
$\sum_{i=0}^{4} N_T =$	2	1	2	4	2
$\sum_{i=1}^{4}(C_i) =$	45	48	59	75	53
$\overline{C_i} =$	11.25	12	14.78	18.75	13.25

Remark:

- EDD yields the minimum, maximum tardiness $(\sum_{i=0}^{4} T_{di} = 6)$, average tardiness $(\overline{[(T]_{di}} = 1.5)$ and number of tardy jobs $\left(\sum_{i=0}^{4}[N_T = 1)]\right)$
- SPT results at smallest mean flow time $(\overline{[(C]_i} = 11.25)$ or completion time $\left(\sum_{i=1}^{4}(C_i) = 45\right)$.

6. Branch and Bound Algorithm (Exact Algorithm)

Branch and bound (Exact) algorithm are most successful approach of exact algorithms. Branch and bound algo-

rithm are a powerful enumerative method, which can find the global optimal solutions for many combinatorial optimization problems. Land & Doig (1960) was the first who developed the Branch and Bound algorithm for discrete programming. As implies by their name branching as well as bounding are two tools which are required for branch and bound algorithm.

- Branching: Branching procedure is splitting procedure. In this procedure we describe how to split the problem (Set A of candidates) into two or more sub problems {Subsets like $A(1), A(2) \ldots \ldots$} such that, their union cover the set of candidates (A) or return the main problems. This procedure is known as branching.
- Bounding: The main aim of bounding procedure is to computing the UB (upper bound) and LB (lower bound). These bounds are an essential tool in a B & B (branch and bound) methodology. The upper or lower bounds are applied to short the search space or computational efforts. The formula applied to calculate the LB is pertinent to objective function of the scheduling problem and we branched only for those nodes which has minimum lower bound value.

6.1 Developed Branch and Bound (Exact) Algorithm for Tardiness Scheduling Problem with Single Machine

Step 1: We start from "0" level. At level zero, the root node will be placed with all n empty sequenced jobs. In this level we start with no job sequenced. It is designated by $(*,*,*,* \cdots \ldots \ldots \ldots .*)$ where $\square$ represent the empty jobs of n jobs set.

$$A(0) = (*,*,*,* \cdots \ldots \ldots \ldots .*)$$

Where $B(0)$ represent a single machine sequencing containing n jobs.

Step 2: At level 1, there will be n number of nodes. Each node will contain a partial sequence of jobs. The problem $A(0)$ divide n sub problems, {A(1), $A(2), A(3)$ ……., $A(n)$} by assigning the last position in sequence. We move from $A(0) = (*,*,*,* \cdots \ldots \ldots \ldots .*)$ to $\{A(1) = (1,*,*,* \cdots \ldots .*), A(2) = (2\ *,*,*,* \cdots \ldots .*)$, ….., A$(n) = (n\ *,*,*,* \cdots \ldots \ldots \ldots .*)\}$.

At this level, we assign or schedule the first job for first position and calculate the Lower Bound like $\{LB(1), L\ B(2), LB(3)$ ……., $LB(n)\}$ for one scheduled job and select that node which has minimum lower bound value from it. If some nodes have equal minimum lower bound value, then we branch from all these nodes.

Step 3: At this level, we branch the minimum value node to find the lower bound of two scheduled jobs and again select the minimum lower bound value from it. Hence, the n nodes are to be branched in the (n-1) available branching like {$A(1), A(2), A(3)$……., A(n)} to{$[A(12), (13), \ldots . (1n)\ ], [A(21), B(23), \ldots . A(2n)]$} ……..,…….}and calculate the lower bound for each partial schedule.

Step 4: Branch the minimum lower bound value node to find the lower bound of three schedule jobs and again select the minimum value from it

Step 5: Until all the jobs are sequenced or assigned, we will continue the process and find the optimal sequence by branching the schedule jobs.

6.2 Condition for Calculating Lower Bound

We check the condition, whether unscheduled jobs are eligible for calculating lower bound or not. For any unscheduled job j_u, its actual tardiness T_{j_u} cannot be less than, to or equal to artificial tardiness $\{max[(C_{ju} - d), 0]\}$ or $T_{j_{ud}}$ (Tardiness with common due date)

$$T_{j_u} \geq max[(C_{ju} - d), 0]$$

$$T_{j_u} \geq T_{j_{ud}}$$

Where d is the largest due date among the unscheduled jobs and the unscheduled jobs have a common due dates considering an artificial problem

$$d = \max(d_{r+1}, d_{r+2}, \ldots \ldots \ldots \ldots \ldots d_n)$$

If the condition is not satisfied, the tardiness of unscheduled job j_u is less than to,

$$max[(C_{ju} - d), 0]$$
$$\square\{T\square_{\downarrow}(j_{\downarrow}u) \ngeq max[(C_{\downarrow}ju - d), 0] \quad or$$
$$T_{\downarrow}(j_{\downarrow}u) < max[(C_{\downarrow}ju - d), \mathbf{0}]\},$$
$$[\![T]\!]_{j_u} \ngeq T_{j_{ud}} \quad or \quad T_{j_u} < T_{j_{ud}}\}$$

Then its branch is said to fathomed branch and no more branching are possible from fathomed branches.

6.3 Calculating Lower Bound

Consider problem $1\|\ D_{\downarrow}(i\ )\|\sum\cong T_{\downarrow}(di\ )$. First, we check the condition whether job is eligible for a particular position or not for calculating lower bound for any unscheduled jobs. The Gantt Chart between Scheduled & Unscheduled Jobs are presented in Figure 3

For any scheduled or assigned jobs, their actual tardiness is defined as

$$T_{js} = max\{(C_{js} - d_{js}), 0\}$$

For any unscheduled job j_u, its tardiness with common due date,

$$T_{j_{ud}} = \max\left\{(C_{ju} - d), 0\right\}$$

The Tardiness of unscheduled jobs are calculated by applying EDD rule.

{Lower Bound = Tardiness of Scheduled Jobs + Tardiness of Unscheduled jobs (calculated with common due date)}

$$LB = \{(T_{j1} + T_{j2} + \ldots\ldots T_{jr}) + (T_{j(r+1)} + T_{j(r+2)} + \ldots\ldots T_{jn})\}$$
$$LB = \{(T_{js}) + (T_{j_{ud}})\}$$

The Lower bound is applied in a different manner for different scheduling problems. We calculate the lower bound in each step to avoid the complete counting of all the permutations of jobs.

7. Numerical Solved By Branch and Bound Algorithm

We solve the same problem by Branch & Bound Algorithm which we solved by Dispatching Rule.

7.1 Calculating lower bound

7.1.1 Calculating Lower Bound for assigning one Job {LB(1), LB(2), LB(3) and LB(4)}

In the partial Schedule $(1,*,*,*)$ assign the job 1 and sequenced the unscheduled jobs, according to EDD Dispatching Rule. First, we check the conditions for unscheduled jobs.

$$T_{j_u} = T_2 + T_3 + T_4 = \{\max(14-16,0) + \max(9-4,0) + \max(24-18,0)\} = 0+5+6 = 11$$

Artificial total tardiness computed with respect to largest common due dates, $d = \max(d_2, d_3, d_4) = 18$

$T_{ju_d} = \max\left[(C_{ju} - d), 0\right]$, (Total tardiness of unscheduled jobs with common due date)

$$T_{ju_d} = [\{max][(C_3 - d), 0] + \max[(C_2 - d), 0] + \max[(C_4 - d), 0]\}$$
$$[= \{\max](9-18,0) + \max(14-18,0) + \max(24-18,0)\} = 0 + 0 + 6 = 6$$

Where $d = \max(d_3, d_2, d_4) = \max(4, 16, 18) = 18$ (unscheduled jobs)

$$T_{ju} \geq T_{ju_d}$$

Condition for lower bound is satisfied. So we calculate the lower bound LB (1)

$T_{\downarrow}js = 0$ (Tardiness of schedule jobs),
$T_{ju} = 11$ (tardiness of unscheduled jobs),
$T_{ju_d} = 6$ (Artificial Tardiness)

$\mathbf{LB(1)}$ = **tardiness of schedule jobs** + Tardiness of unscheduled jobs with common due date

$$\mathbf{LB(1)} = T_{js} + T_{ju_d} = 0 + 6 = 6$$
$$\mathbf{LB(1)} = 6$$

Similarly, we calculate $\mathbf{LB(2) = 6, LB(3) = 6}$ **and** $\mathbf{LB(4) = 9}$ now select the minimum value of lower bound and branch from this node. There are three nodes $\mathbf{LB(1), LB(2)}$ and $\mathbf{LB(3)}$ of similiar lower bound value. Consequently, we branch from these three nodes.

7.1.2 Calculating lower bound for assigning two jobs like LB(12)

$\mathbf{LB(12)}$, that means we scheduled the job 1 and job 2 and unscheduled jobs are 3 and 4.

First, we check the condition of whether unscheduled jobs (3 and 4) are eligible for calculating lower bound or not.

Actual total tardiness of unscheduled jobs, $(j_{\downarrow}3, j_{\downarrow}4)$

$$T_3 + T_4 = \{\max(9-4,0) + \max(24-18,0)\} = 5+6 = 11$$

Artificial total tardiness computed with respect to largest common due dates,

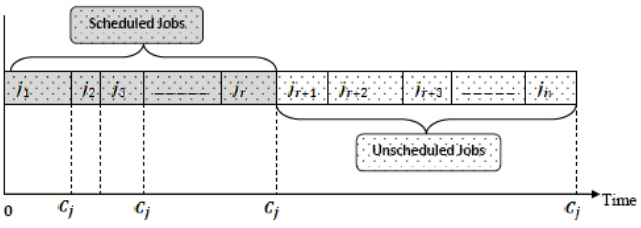

Figure 3. Gantt chart between Scheduled and Unscheduled Jobs.

$$d = \max(d_3, d_4) = 18$$

$$\{\max\{(C_3 - d, 0)\} + \{\max\{(C_4 - d, 0)\} = \max(9 - 18, 0) + \max(24 - 18, 0)\} = 0 + 6 = 6$$

$$T_3 + T_4 \geq \{\max[(C]_3 - d, 0)\} + \{\max[(C]_4 - d, 0)\}$$

$$\geq \{\max(9 - 18, 0) + \max(24 - 18, 0)\}$$

$$T_{ju} \geq T_{ju_d}$$

The Condition is satisfied. So we proceed to calculate the lower bound for partial scheduled $(1, 2, *, *)$

$\mathbf{LB(12) = Tardiness\ of\ schedule\ jobs(1\&2) +}$ Tardiness of unscheduled jobs (3&4) by applying EDD rule with a common due date

$$\mathbf{LB(12)} = T_{js} + T_{ju_d} = \mathbf{0 + 6 = 6}$$

Similarly, we calculate $\mathbf{LB(13) = 11, LB(14) = 11, LB(21) = 7, LB(23) = 8, LB(24) = 16, LB(31) = 6,}$ $\mathbf{LB(32) = 6\ and\ LB(34) = 11}$

Minimum values of the lower bound for scheduling two jobs are $\mathbf{LB(12), LB(31)\ and\ LB(32)}$. So we branch from these nodes. Similarly, we calculate the lower bound for scheduling three jobs.

{

$\mathbf{LB(123) = 16, LB(124) = 25, LB(312) = 6,}$

$\mathbf{LB(314) = 9,\quad LB(321) = 8, LB(324) = 12}$

}

Lower bound for scheduling the jobs (j_3, j_1, j_2) is minimum. So we branch from this node and we obtained optimal sequence as (j_3, j_1, j_2, j_4).

7.2 Computational Result

Numerical solved by Branch & Bound Algorithm is shown in Table 5.

Remark

- We used EDD rule for sequencing unscheduled jobs
- We find the optimal sequence from B&B (j_3, j_1, j_2, j_4) that is same as we obtained using EDD rule
- So we can say that EDD rule gives the optimal solution for total tardiness problems and it works as an exact algorithm.

Table 5. Four jobs single machine scheduling problem with due dates solved by B&B Algorithm

Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(1)
1	8	12	8	0	Scheduled job(j_1) Unscheduled jobs (j_3, j_2, j_4)
3	1	4	9	5	
2	5	16	14	0	
4	10	18	24	6	
					LB(1) = 6
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(2)
2	5	16	5	0	Scheduled job (j_2) Unscheduled jobs (j_1, j_3, j_4)
3	1	4	6	2	
1	8	12	14	2	
4	10	18	24	6	
					LB(2) = 6
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(3)
3	1	4	1	0	Scheduled job(j_3) Unscheduled jobs (j_1, j_2, j_4)
1	8	12	9	0	
2	5	16	14	0	
4	10	18	24	6	
					LB(3) = 6

(*Continued*)

Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(4)
4	10	18	10	0	Scheduled job(j_4)
3	1	4	11	7	
1	8	12	19	7	Unscheduled jobs
2	5	16	24	8	(j_1, j_2, j_3)
					LB(4) = 9
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(12)
1	8	12	8	0	Scheduled job(j_1, j_2)
2	5	16	13	0	
3	1	4	14	10	Unscheduled jobs
4	10	18	24	6	(j_3, j_4)
					LB(12) = 6
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(13)
1	8	12	8	0	Scheduled job (j_1, j_3)
3	1	4	9	5	
2	5	16	14	0	Unscheduled jobs
4	10	18	24	6	(j_2, j_4)
					LB(13) = 11
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(14)
1	8	12	8	0	Scheduled job(j_1, j_4)
4	10	18	18	0	
3	1	4	19	15	Unscheduled jobs
2	5	16	24	8	(j_3, j_2)
					LB(14) = 11
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(21)
2	5	16	5	0	Scheduled job (j_2, j_1)
1	8	12	13	1	
3	1	4	14	10	Unscheduled jobs
4	10	18	24	6	(j_3, j_4)
					LB(21) = 7
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(23)
2	5	16	5	0	Scheduled job(j_2, j_3)
3	1	4	6	2	
1	8	12	14	2	Unscheduled jobs
4	10	18	24	6	(j_1, j_4)
					LB(23) = 8

(*Continued*)

Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(24)
2	5	16	5	0	Scheduled job $(j_1 2, j_1 4)$
4	10	18	15	0	
3	1	4	16	12	Unscheduled jobs $(j_1 3, j_1 1)$
1	8	12	24	12	
					LB(24) = 16
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(31)
3	1	4	1	0	Scheduled job$(j_1 3, j_1 1)$
1	8	12	9	0	
2	5	16	14	0	Unscheduled jobs $(j_1 2, j_1 4)$
4	10	18	24	6	
					LB(31) = 6
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(31)
3	1	4	1	0	Scheduled job $(j_1 3, j_1 2)$
2	5	16	6	0	
1	8	12	14	2	Unscheduled jobs $(j_1 1, j_1 4)$
4	10	18	24	6	
					LB(32) = 6
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(34)
3	1	4	1	0	Scheduled job$(j_1 3, j_1 4)$
4	10	18	11	0	
1	8	12	19	7	Unscheduled jobs $(j_1 1, j_1 2)$
2	5	16	24	8	
					LB(34) = 11
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(123)
1	8	12	8	0	Scheduled job$(j_1, j_2\ j_1 3)$
2	5	16	13	0	
3	1	4	14	10	Unscheduled jobs $(j_1 4)$
4	10	18	24	6	
					LB(123) = 16
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(124)
1	8	12	8	0	Scheduled job$(j_1 1, j_1 2, j_1 4)$
2	5	16	13	0	
4	10	18	23	5	Unscheduled jobs $(j_1 3)$
3	1	4	24	20	
					LB(124) = 25

(*Continued*)

Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(312)
3	1	4	1	0	Scheduled job $(□j_↓3, j□_↓1, j_↓2)$
1	8	12	9	0	
2	5	16	14	0	Unscheduled jobs $(j_↓4)$
4	10	18	24	6	
					LB(312) = 6
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(314)
3	1	4	1	10	Scheduled job$(j_↓3, j_↓1, j_↓4)$
1	8	12	9	0	
4	10	18	19	1	Unscheduled jobs $(j_↓2)$
2	5	16	24	8	
					LB(314) = 9
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(321)
3	1	4	1	0	Scheduled job$(j_↓3, j_↓2, j_↓1)$
2	5	16	6	0	
1	8	12	14	2	Unscheduled jobs $(j_↓4)$
4	10	18	24	6	
					LB(321) = 8
Jobs (j_i)	Processing time (p_i)	Due dates (D_i)	Completion time (C_i)	Tardiness T_{di}	Lower Bound LB(324)
3	1	4	1	0	Scheduled job$(j_↓3, j_↓2, j_↓4)$
2	5	16	6	0	
4	10	18	16	0	Unscheduled jobs $(j_↓1)$
1	8	12	24	12	
Optimal sequence $(j_↓3, j_↓1, j_↓2, j_↓4)$					LB(324) = 12
Jobs (j_i)	Processing time (m_{i1})	Due dates (D_i)	Completion time (C_i)	Tardiness $T_{di} = \{\max[(C]_i - D_i, 0)\}$	
3	1	4	1	0	
1	8	12	9	0	
2	5	16	14	0	
4	10	18	24	6	
	$\sum_{i=1}^{4}(C_i) = 48$			$\sum_{i=1}^{4}(T_{di}) = 6$	

7.3 Branching Tree

We show our problem in a branching tree structure where each node is the partial schedule. The lower bound values for total tardiness of jobs (objective function) are calculated to determine the best partial schedule node to the branch. We selected to that node that which has the least Lower Bound (LB) value and branch from this node. This operation is executed again and again whenever all the jobs are scheduled by branching from least LB. If the condition of the LB is not satisfied for any node than the branching tree

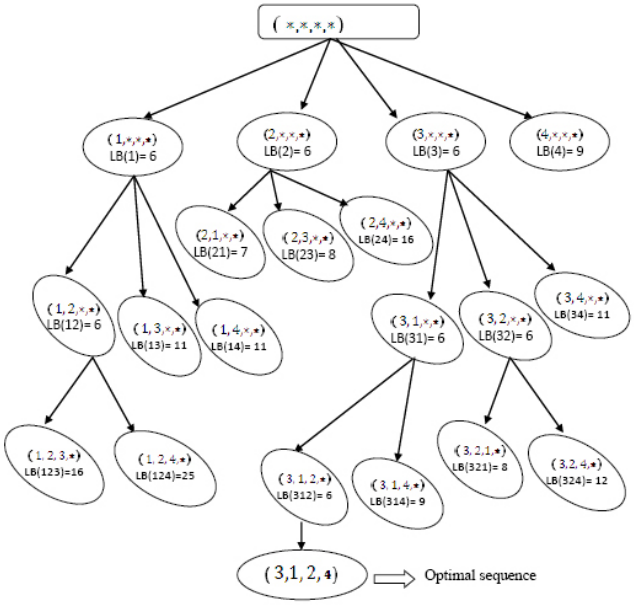

Figure 4. This figure shows the process of lower bounds in the form of Branching Tree.

is fathomed and no more branching is possible from this node. The branching tree is presented in Figure 4.

8. Conclusion and Future Research

We have compared the performances of all of the dispatching rules with the help of numerical illustrations and selected the best dispatching rule for tardiness problem. Comparative study shows that the EDD rule gives the best result to optimize the Total Tardiness / Average tardiness and number of total tardy jobs as compared to other dispatching rules, but there is no guarantee of dispatching rules to give the optimum solution, hence we have developed an exact algorithm (B&B) combining with EDD rule that has produced the optimal sequence for the single machine scheduling problem. In addition, We also solved the same numerical problem by B&B algorithm. We compared the B&B algorithm with Dispatching rules with the help of numerical illustration and result shows that B&B algorithm and EDD dispatching rule give the same result or same sequence which means that EDD rule gives the optimal sequence or optimal solution because the B&B algorithm (exact algorithm) that solves the problem to certain optimality. In the study, we concluded that EDD rule works as an Exact Algorithm for minimization of total (average) tardiness problems or gives the optimal solution for tardiness problems. Computational results show that our approach is effective in solving tardiness problems.

For future research, this study may further extend by considering various parameters like weighted tardiness, Setup Times, breakdown effect, tardiness and earliness cost etc. meta heuristics(like Genetic Algorithm, Ant Colony Optimization, Simulated Annealing, Tabu Search etc..) or hyper heuristics approach can also be used for solving these types of problems.

9. References

1. Sen T, Gupta SK. A state-of-art survey of static scheduling research involving due dates. OMEGA. The Int J of Manag Sci. 1984; 12(1):63–76.
2. Abdul-Razaq TS, Potts CN, Van WLN. A survey of algorithms for the single machine total weighted tardiness problem. Discrete Appl Math. 1990; 26:235–53
3. Koulamas C. The total tardiness problem: review and extensions. Oper Res. 1994; 42(6):1025–41.
4. Tyagi N, Abedi M, Varshney RG. 2013. A preemptive scheduling and due date assignment for single-machine in batch delivery system. Proceeding of CACCS, India. 2013; 60–4.
5. Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximating in deterministic sequencing and scheduling: A survey. Ann Discrete Math. 1979; 4:287–326.
6. Jackson JR. Scheduling a production to minimize maximum tardiness. Research Report 43, Manag Sci Res Project, University of California at Los Angeles. 1955
7. Haupt R. A Survey of Priority Rule-Based Scheduling. OR Spectgrum. 1989; 11(3):3–16.
8. Chiang TC, Fu LC. Using dispatching rules for job shop scheduling with due date-based objectives. Int J of Pro Res. 2007; 45(14):3245–62.
9. Naughton MR. Scheduling with deadlines and loss functions. Mngt Sci. 1959; 6(1): 1–12.
10. Conway RW. Priority dispatching and job lateness in a job shop. J of Ind Eng. 1965; 16(5):228–37.
11. Baker KR. Introduction to Sequencing and Scheduling. Wiley: New York. 1974.
12. Morton TE, Pentico DW. Heuristic Scheduling Systems. John Wiley and Sons. New York, NY. 1993.
13. Du J, Leung JY-T. Minimizing total tardiness on one machine is NP-hard. Math of Oper Res. 1990; 15(3):483–95.
14. Moore JM. An n job, one machine sequencing algorithm for minimizing the number of late jobs. Mngt Sci. 1968; 15(6):102–9.
15. Emmons H. One machine sequencing to minimize certain functions of job tardiness. Oper Res. 1955; 17:701–15.

16. Land AH, Doig G. An automatic method of solving discrete programming problems. ECONOMETRICA. 1960; 28(3): 497–520.
17. Brown APG, Lomnicki ZA. Some applications of the branch and bound algorithm to the machine scheduling problem. Oper Res Quarterly. 1966; 17(2):173–80.
18. Potts CN, Wassenhove VLN. A decomposition algorithm for the single machine total tardiness problem. Oper Res Letters. 1982; 1(1):177–81.
19. Chu C. A branch-and-bound algorithm to minimize total tardiness with different due dates. Navl Res Logi. 1992; 39(9):265–83.
20. Hirakawa Y. A quick optimal algorithm for sequencing on one machine to minimize total tardiness. Int J Prod Econ. 1999; 549–55.
21. Kondakci S, Kirca O, Azizoglu M. An efficient algorithm for the single machine tardiness problem. Int J of Pro Eco. 1994; 36(2):213–9.
22. Biskup D, Piewitt W. A note on An efficient algorithm for the single machine tardiness problem. Int J of Prod Eco. 2000; 66(3):287–92.
23. Szwarc W, Croce DF, Grosso A. Solution of the single-machine total tardiness problem. J of Scheduling. 1999; 26(2-3):55–71.
24. Szwarc W, Grosso A, Croce F. Algorithmic paradoxes of the single machine total tardiness problem. J of Scheduling. 2001; 4(2):93–104.
25. Tansel BC, Kara BY, Sabuncuoglu I. An efficient algorithm for the single machine total tardiness problem. IIE Transactions. 2001; 33(8):661–74.
26. Baker KR, Scudder GD. Sequencing with earliness and tardiness penalties- A review. Oper Res. 1990; 38(1):22–36
27. Tian ZJ. On the single machine total tardiness problem. Eur J of Oper Res. 2005; 59(3):843–6.
28. Koulamas C, Kyparisis GJ. Single machine scheduling with release times, deadlines and tardiness objectives. Eur J of Oper Rese. 2001; 133(1):447–53.
29. Baptiste P. Scheduling equal-length jobs on identical parallel machines, Discrete Appl Math. 2000; 103(1): 21–32.
30. Koulamas C. The single machine total tardiness scheduling problem: Review and extensions. Eur J of Oper Res. 2010; 202(1): 1–7.