New Approximation Algorithms for Minimizing Number of Tardy Jobs on Single Machine with Release Dates

A.O. Akinrinde[*1], E.O. Oyetunji[2], A.E. Oluleye[3]

[1, 3]Department of Industrial and Production Engineering
University of Ibadan, Nigeria

[2]Department of Mechanical Engineering
Lagos State University, Nigeria

[1]tunjiakinrinde@yahoo.com; [2]eoyetunji@yahoo.com; [3]aoluleye@yahoo.com

***Abstract*- This paper deals with the scheduling problem of minimizing number of tardy jobs on a single machine with release dates. The problem is considered as NP-Hard, because it is difficult to solve and no known polynomial bounded algorithm exist that guarantees optimal solution. Therefore, four approximation algorithms (JB1, JB2, JB3 and JB4) were proposed to solve the problem. The proposed algorithms were compared with two heuristics (DAU and EOO) that were proposed previously for this same problem. Twenty-two problem sizes ranging from three to five hundred jobs with fifty different simulated problems for each problem size, which gave one thousand, one hundred problems were solved. The four algorithms were coded using Microsoft visual basic 6.0 while statistica 8.0 was used to analyze the results. Performance evaluation was based on both effectiveness and efficiency. Experimental results were provided while appropriate recommendations were made.**

Keywords- Scheduling; Single Machine; Tardy Jobs; Heuristic; Performance Ratio

I. INTRODUCTION

Scheduling is one of the most widely researched areas of operational research, which is largely due to the rich variety of different problem types within the field [19]. Scheduling is concerned with allocation of scarce resources to tasks under certain constraints with the aim of fulfilling specific performance objectives which revolve around time. This is applicable in different sectors, ranging from transportation, banking, health, etc [17]. Scheduling objectives which are also called performance measures are criteria by which the performances of systems are measured. Scheduling objectives could be based on completion time, due date and flow time, etc. Criteria based on due dates involve determining whether a particular sequence of jobs will make a job late or not. A job is said to be late if the job is completed after the allotted time that is designated for the job. Penalties could be incurred on jobs that are late and this could be in form of cost, hazard or loss of good will [10]. Consider a banking hall; if it takes too long to service a customer, it may cost the customer, lost of business or other important thing. Hence he may decide not to employ the services of the bank anymore, thereby leading to loss of customer.

Scheduling problems are combinatorial in nature. The number of possible schedules that is obtainable in a case of n jobs and m machines is $(n!)^m$, [18]. If the values of n and m are relatively small, the problem is solvable but every increase in the values of n and m makes the problem difficult to solve, even it might take a fast computer thousands of years to come up with a solution. For instance, scheduling five jobs on five machines will yield; $(5!)^5 = 2.4 \times 10^{10}$ schedules which will take 69.12 hours for a computer evaluating at a one hundred-thousand per second to come up with a solution. The increment of both the job and machine to six will yield 1.003×10^{20} schedules, which will take 31806865 years for the same computer evaluating at one hundred-thousand per second. This is outrageous because the duration is not within the life time of man performing the schedule. Hence, this is not viable as the time and cost of obtaining the best solution is too exorbitant compared to its importance. Thus, there is need for a method that can generate a fair solution if not optimal solution within a short time.

Therefore, the objective of this paper is to explore the scheduling problem of minimizing the number of tardy jobs on a single machine with release dates. Minimizing the number of tardy jobs criterion is very important in scheduling because of the penalties attached to tardy jobs. Approximation algorithms are adopted for solving this problem because they give solutions faster than other methods, hence, more job size problems could be solved within a reasonable computational time. To the best of my knowledge, very few heuristics have been proposed to solve this particular problem.

II. LITERATURE REVIEW

Jackson (1955) showed that a sequence which minimizes maximum lateness is given by earliest due date rule (EDD rule) in which the jobs are sorted in non-decreasing order of their due dates dj. Moore (1968) proposed an $O(n \log n)$ polynomial algorithm for the most general problem, $[1||\sum U_i]$ by applying the EDD rule to sequence the jobs [16]. The dynamic programming algorithms proposed by Kise et al. (1978) proved that when release dates and due dates are similarly ordered $(r_1 \leq r_2 \leq r_3 \text{---} \leq r_n \rightarrow d_1 \leq d_2 \leq d_3 \text{---} \leq d_n)$ the problem is solvable in $O(n^2)$ polynomial time [14].

Carlier (1981) proposed an $O(n^3 \log n)$ polynomially bounded algorithm for the problem with equaling processing times: $[1 \mid r_i, p_i = p \mid \Sigma U_i]$ [4]. Lawler (1990) proposed a procedure bounded by $O(n^5)$ dynamic programming algorithm with distinct release dates and job preemption is allowed, $[1|rj,pmtn| \Sigma U_i]$ [13]. The complexity for the problem with similarly ordered release and due dates was reduced to $O(n \log n)$ by Lawler (1994) using a modification of the Moore (1968) algorithm.

By relaxing binary constraints in mixed integer linear programming formulations (MILP) which was emerged due to the works of Lasserre and Queyranne (1992), Dauzere-perez (1995) and Dauzere-perez (1997) obtained lower bounds calculation for the problem $[1 \mid pmtn, r_i \mid \sum U_i]$ [6-8, 11]. Due to the size of MILP program, no more than ten jobs could be considered and therefore, it is not suitable to implements it branch and bound procedures.

Bapstiste et al. (1998) interpreted the problem as a constraint satisfaction problem, he developed constraint propagation techniques which were embedded into a branch and bound procedure. Baptiste (1999) proposed algorithm that is polynomially bounded by $O(n^4)$ and thus provides the lowest complexity for the problem of $[1|rj,pmtn| \Sigma U_i]$ [1]. Bapstite et al. (2003) consequently used elimination rules and dominance properties in order to quicken the search procedure [2].

Four different branch and bound procedures for $[1 \mid r_i \mid \sum U_i]$ attract attention. Dauzere-perez and Sevaux (2004) suggested a branch and bound method upon the notion of the master sequence, i.e. a sequence from which it is known that it contains at least one optimal solution [9]. Chrobak et al. (2004) proved that the algorithm proposed by Carlier in 1981, may produce suboptimal results and they presented $O(n^5)$ dynamic programming algorithm for the problem having equal processing time [5]. Briand et al. (2006) introduced $O(n^5)$ algorithm which showed that an approach to a lower bound is to relax due dates such that the time manipulation of $(d_i - r_i)$ becomes nested; $[r_1, d_1] \subseteq [r_2, d_2] \subseteq -[r_n, d_n]$ [3]. For this case $([1| r_i, nested \mid \sum U_i])$. Recently, M'Hallah and Buffin (2007) published a branch and bound approach for the weighted case, which is also capable of solving the non-weighted case [15]. They formulated $[1| r_i \mid \sum U_i]$ as a linear program and used surrogate relaxation to obtain a multiple choice knapsack problem for which branch and bound algorithms are well-known.

Oyetunji and Oluleye (2008) developed EOO heuristic for $[1 \mid r_i \mid \sum U_i]$ problem [18]. The heuristic solved problem size ranging from three to five hundred jobs with 22 different problem sizes with each problem sizes replicated fifty times to give one thousand, one hundred problems in total. Result from EOO was compared that of DAU heuristics, EOO performed better than DAU both in terms of effectiveness and efficiency.

The problem of minimizing number of tardy jobs on single machine with release dates $[1 \mid r_i \mid \sum U_i]$ has been proven to be NP-Hard problem by Lenstra, Rinnooy and Brucker (1977) [12], thus four heuristics were proposed to solve the problem. To the best of our knowledge, very few heuristics have been proposed to solve this particular problem. DAU heuristic by Dauzere-Perez (1995) [6]and EOO heuristic by Oyetunji and Oluleye (2008) [18] are some of the heuristics that have been proposed to solve the scheduling problem of minimizing number of tardy jobs on a single machine with release dates. Therefore, the two heuristics were selected for evaluation purposes. Performance evaluation was based on both effectiveness and efficiency.

III. THE PROBLEM

For a single machine scheduling problem, the followings are given:

A set of n jobs; $J_1, J_2, J_3, \ldots, J_n$

The processing time of each job; P_i. Processing time (P_i) is the time taken for job i to be processed on a particular machine.

The release or ready time of each job; r_i. Release date (r_i) is the time a job i is ready to be processed.

The due date of each job; d_i. Due date (d_i) is the time promised that the job will be completed.

Completion time (C_i) is the time the processing of a job is completed. If the job is a multi-operation job, the completion time of the job is the time processing of the last operation of the job is completed. Hence, the completion time of a job is the sum of the processing time p_i and release date r_i.

$$C_i = p_i + r_i \tag{1}$$

The lateness of a job i is defined as $L_i = C_i - d_i$, thus when the completion time of job C_i is greater than its due date d_i; $(C_i > d_i)$, the job is said to be late. Lateness is positive when the job is late while it is negative when the job is early; if the completion time of job C_i should be less than its due date d_i, $(C_i < d_i)$, then is said to be early.

Tardiness of job i is defined as

$$T_i = max\,(C_i - d_i, 0), \tag{2}$$

$$T_i = max\,(L_i, 0) \tag{3}$$

The difference between tardiness and lateness is that tardiness is never negative.

Thus, we define

$$U_i = \begin{Bmatrix} 1; if\, C_i > d_i \\ 0; otherwise \end{Bmatrix} \quad (4)$$

Therefore, define the number of tardy jobs (N_T) as

$$N_T = \sum_{i=1}^{n} U_i \quad (5)$$

We assumed that the problem is static and deterministic (i.e. all the parameters (number of jobs, processing times, due dates, release dates) are all fixed and known with certainty.

Thus, the problem being explored in this paper can be represented as $1 \mid r_i \mid \sum U_i$ using the three dimensional notation of Graham et al (1979).

IV. PROPOSED HEURISTICS

The four heuristics developed to solve the problem of minimizing number of tardy jobs are discussed below.

A. JB1 Heuristic

The first heuristic developed in this paper is labeled JB1. The JB1 heuristic first arranges jobs in the non-decreasing order of due date (i.e. according to EDD rule). It is known that the time interval between the due date and the release date is called allowance. This is the maximum time a job can stay in the shop without being tardy. Considering the fact that for a job to be early, lateness is negative, thus lateness is

$$L_i = d_i - C_i$$

Since $C_i = p_i + r_i$

Lateness $L_i = d_i - (p_i + r_i)$

$$L_i = d_i - p_i - r_i$$

$$L_i = (d_i - r_i) - p_i = a_i - p_i$$

Where, $a_i = (d_i - r_i)$

Therefore, it is conjectured that the larger the allowance, the more tardy the job could be. Thus, whenever a tardy job is met, the job with the largest $a_i = (d_i - r_i)$ is removed to be scheduled later. If a decision of jobs with the same allowance is to be taken, the job with the smallest due date is preferred to be removed.

1) JB1 Steps:

Step 1: Initialization

Set 1 = $\{J_i\}$ where i ranges from 1 to n for a given set of jobs.

Set 2 = { }

Step 2: Arrange the jobs according to non-decreasing order of due dates of the jobs in set 1.

Step 3: Compute allowance $a_i = (d_i - r_i)$ of all the jobs.

Step 4: Check if the jobs are tardy $(C_i > d_i)$ starting from the job with smallest due date at time $s_i \geq C_i$.

Step 5: If tardy job is found, remove the job with largest allowance a_i from set 1 to set 2, otherwise schedule the job and compute completion time C_i.

Step 6: Step 4 is repeated for the next job until $i = n$, then go to step 6.

Step 7: The sequence of the jobs is formed by appending jobs remaining in set 1 to jobs in the set 2. The number of tardy jobs is the number of jobs in set 2.

Step 8: Stop.

B. JB2 Heuristic

This second proposed heuristic arose from the analysis of allowance carried out under JB1 heuristic. The JB2 heuristic first computes allowance ($a_i = (d_i - r_i)$) for each job and then schedules the jobs according to the non-decreasing order of allowance. Whenever a tardy job is met, the job with largest processing time is removed. When the decision of removing either of jobs

with the same processing time is encountered, the job with the smallest due date is removed.

1) JB2 Steps:

Step 1: Initialization

Set 1 = $\{J_i\}$ where i ranges from 1 to n for a given set of jobs.

Set 2 = {}

Step 2: Compute allowance $a_i = (d_i - r_i)$ of the jobs and arrange the jobs according to non-decreasing order of allowance.

Step 3: Check if the jobs are tardy $(C_i > d_i)$ starting from the job with smallest allowance at time $s_i \geq C_i$.

Step 4: If tardy job is found, remove the job with largest processing time from set 1 to set 2, otherwise schedule the job and compute completion time C_i.

Step 5: Step 3 is repeated for the next job until $i = n$, then go to step 5.

Step 6: The sequence of the jobs is formed by appending jobs remaining in set 1 to jobs in the set 2. The number of tardy jobs is the number of jobs in set 2.

Step 7: Stop.

C. JB3 Heuristic

JB3 heuristic is based on the fact that it is expected that there should be reasonable difference between the due date of a particular job and its completion time assuming no job is done before it. Hence, if the jobs that have small difference between their due dates and the completion times are preferable remove from the schedule, this will lessen the number of tardy jobs.

The JB3 heuristic first arranges jobs according to the non-decreasing order of due date (i.e. according to EDD rule). Then, the sum of processing time and release date (which is the completion time) for each job is computed. Whenever a tardy job is met, the job with the highest individual completion time is removed to be scheduled later. When the decision of removing either of jobs with the same processing time is encountered, the job with the largest processing time is removed.

1) JB3 Steps:

Step 1: Initialization

Set 1 = $\{J_i\}$ where i ranges from 1 to n for a given set of jobs.

Set 2 = {}

Step 2: Arrange the jobs according to non-decreasing order of due dates of the jobs in set 1.

Step 3: Compute individual completion time $C_i = p_i + r_i$ of all the jobs.

Step 4: Check if the jobs are tardy $(C_i > d_i)$ starting from the job with smallest due date at time $s_i \geq C_i$.

Step 5: If tardy job is found, remove the job with largest individual completion time $(p_i + r_i)$ from set 1 to set 2, otherwise schedule the job and compute completion time C_i.

Step 6: Step 4 is repeated for the next job until $i = n$, then go to step 6.

Step 7: The sequence of the jobs is formed by appending jobs remaining in set 1 to jobs in the set 2. The number of tardy jobs is the number of jobs in set 2.

Step 8: Stop.

D. JB4 Heuristic

This algorithm has similar principle with first-in-first-out (FIFO). The jobs that arrive the shop earlier are preferred to jobs that come later. Therefore, the release date happens to be the criteria for removing jobs in the schedule. The jobs are arranged in order of non-decreasing order of due date (i.e. according to EDD rule). Whenever a tardy job is met, the job with the largest release date r_i is removed to be scheduled later. If a decision of jobs with the same release date is to be removed, the job with the largest processing time is preferred to be removed.

1) JB4 Steps:

Step 1: Initialization

Set 1 = $\{J_i\}$ where i ranges from 1 to n for a given set of jobs.

Set 2 = { }

Step 2: Arrange the jobs according to non-decreasing order of due dates of the jobs in set 1.

Step 3: Check if the jobs are tardy $(C_i > d_i)$ starting from the job with smallest due date at time $s_i \geq C_i$.

Step 4: If tardy job is found, remove the job with largest release date from set 1 to set 2, otherwise schedule the job and compute completion time C_i

Step 5: Step 3 is repeated for the next job until $i = n$, then go to step 6.

Step 6: The sequence of the jobs is formed by appending jobs remaining in set 1 to jobs in the set 2. The number of tardy jobs is the number of jobs in set 2.

Step 7: Stop.

V. SELECTED HEURISTICS

Two heuristics that addressed the problem of minimizing number of tardy jobs on a single machine with release dates were selected from the literature to serve as comparative purpose to the proposed heuristics. The two heuristics are: EOO and DAU.

A. EOO Heuristic

EOO heuristic (which was proposed by Oyetunji and Oluleye, 2008) [18] makes use of the fact that for jobs not to be tardy, priority must be given to the due dates of the jobs. The heuristic considers the next job that is due out of all the jobs to be scheduled and checks if scheduling the job at that time would make the job tardy. If the job is likely to be tardy, the job will not be scheduled but will be added to the set of tardy jobs but if not tardy, the job is scheduled.

1) EOO Steps:

Step 1: Initialization

$Job\_Set1 = \{J_i\}_{i=1}^{n}$ This is the set of given jobs

$Job\_Set2 = \{\}$ This is the set of jobs tested and found to be early

$Job\_Set3 = \{\}$ This is the set of jobs tested and found to be tardy

$i = 1$

Step 2: Select the job with the lowest due date from *Job_Set1*.

Step 3: Check if scheduling the job in step 2 in the i^{th} position will make the job tardy or not.

Step 4: If the job will be tardy, do not schedule it, but add the job to *Job_Set3*; otherwise schedule it and add the job to *Job_Set2*. Remove the job from *Job_Set1*.

Step 5: $i = i + 1$, if *Job_Set1* is not empty then go to step 2; otherwise go to step 6.

Step 6: The required sequence of jobs is formed by appending jobs in *Job_Set2* to jobs in *Job_Set3*. The number of jobs in *Job_Set3* is the number of tardy jobs.

Step 7: Stop.

B. DAU Heuristic

This heuristic (which was proposed by Dauzere-perez, 1995) divides *J*, the set of jobs to be scheduled, into three *O*, *L* and *F*, where

O is a set of jobs scheduled on time;

L is a set of tardy jobs;

F is a set of outstanding free jobs which are not scheduled at a special point during the heuristic.

Choose the job with the smallest due date among the jobs arrived at time *t* from set *F*, schedule it, if it is late and add it to set *L*, otherwise add it to set *O*.

2) DAU Steps:

Step 0: Initialization

$$\text{Set } t = \min_{J_i \in J} r_i,\ O=L=\varphi, \text{ and } F = J$$

Step 1: While $F \neq \varphi$, do choose the job J_j with smallest due date, such that $r_i \leq t$

Step 2: $O = O \cup \{J_j\}$, $F = F - \{J_j\}$, $S_j = t$ and $C_j = s_j + p_j$

Step 3: If $C_j > d_j$ then Search $J_i \varepsilon O$ such that $C(O - \{J_j\}) = \min_{J_t \varepsilon J} r_i\, C(O - \{J_j\})$

$$O = O - \{J_j\},\ L = L \cup \{J_j\},\ \text{update } s_k$$

$$C(O) = s_k + p_k \text{ where } J_k \text{ is the last job.}$$

Step 4: If there are jobs in L such that $r_i \leq s_k$ and $s_k + p_k \leq d_i$ then choose the one with the smallest processing time to be J_l. If J_l exists and $p_l < p_k$, then

$$O = O - \{J_k\},\ O = O \cup \{J_l\},\ L = L \cup \{J_j\},\ L = L \cup \{J_k\}$$

Update s_l, $C(O) = s_l + p_l$

Step 5: $t = max\{C(O), \min_{J_t \varepsilon F} r_i\}$

Step 6: If set F is not empty, go back to step 1: otherwise proceed to step 8.

Step 7: Form by appending jobs in O to jobs in L. The number of jobs in L is the number of tardy jobs.

Step 8: Stop.

VI. DATA ANALYSIS

The proposed and selected heuristics were coded using Microsoft visual basic 6.0. The program was also used to randomly generate single machine problems. The job size ranges between 3 and 500 inclusively ($3 \leq n \leq 500$), 22 different job sizes were explored, 50 different samples of each job size were replicated, to make a total of 1,100 problems. Processing time was non-uniformly generated between 1 and 100 inclusively ($1 \leq p_i \leq 100$) while the release date was also randomly generated between 0 and p_i inclusively ($0 \leq r_i \leq p_i$) for each job i. The due date ranges from $(r_i + p_i)$ to $(r_i + 2p_i)$. The data were generated with the aid of visual basic random number generator.

The Statistica 8.0 (statistical software) was used to carry out data analysis. Statistical 8.0 was used to compute: mean test, approximation ratio, percentage ranking and T-test.

VII. RESULT AND DISCUSSION

The mean value of the number of tardy jobs obtained from the various solution methods and job sizes is shown in Table 1. It is observed that averagely, EOO gave the lowest mean value of number of tardy jobs, follow by JB2, JB3, DAU, JB1, and JB4 in that order. This confirms the results of Oyetunji and Oluleye (2008) in respect of EOO heuristic [18].

TABLE 1 MEAN VALUE OF NUMBER OF TARDY JOBS OBTAINED FROM VARIOUS SOLUTION METHODS

Job Size *n*	JB1	JB2	JB3	JB4	EOO	DAU
3	0.84	0.82	0.82	0.86	0.8	0.85
4	1.48	1.44	1.32	1.42	1.4	1.45
5	2.32	2.38	2.3	2.42	2.3	2.4
6	2.82	2.8	2.74	2.98	2.8	2.86
7	3.84	3.84	3.7	4.06	3.74	3.89
8	4.46	4.42	4.38	4.66	4.38	4.47
9	5.32	5.12	5.08	5.56	5.08	5.18
10	6.18	6.02	6.04	6.46	5.98	6.06
12	7.92	7.86	7.84	8.52	7.74	7.89
15	10.24	10.16	10.02	11.22	10.02	10.22
20	14.68	14.52	14.58	15.92	14.48	14.76
25	19.76	19.28	19.58	20.92	19.48	19.86
30	24.02	24.04	24.08	25.74	24	24.46
40	33.94	33.64	33.78	35.58	33.56	34.2
50	43.44	43.06	43.26	45.58	42.92	43.74
60	52.5	52.86	53.18	55.28	52.76	53.77
80	72.16	71.86	72.4	74.98	71.78	72.45
100	92.56	91.78	92.3	95.24	91.56	92.4
150	142.08	141.04	141.88	144.86	141	142.09
200	191.68	190.6	191.64	194.4	190.3	191.92
300	291.26	289.84	291.02	293.88	289.56	291.07
500	491.28	489.32	490.86	492.94	488.7	490.99

Further analysis (i.e. performance ratio) was carried out on heuristics. Fig.1 shows the visual difference in the performance ratio of the proposed heuristics (JB1, JB2, JB3 and JB4) compared to DAU while Fig. 2 shows the performance ratio of the proposed heuristics compared to EOO. It is observed that JB2 and JB3 did better than DAU heuristic for all the job sizes, while JB1 did better in most cases and worse in few cases but did slightly better than DAU heuristic in average. JB4 performed worse than DAU heuristic (Fig. 1). Averagely, none of the developed heuristics was able to do better than EOO, but in some few cases (job sizes 4 to 9), JB3 did better than EOO heuristic. Generally, JB2 was the closest to EOO heuristic, followed by JB3, while JB4 performance is the worst (Fig. 2).

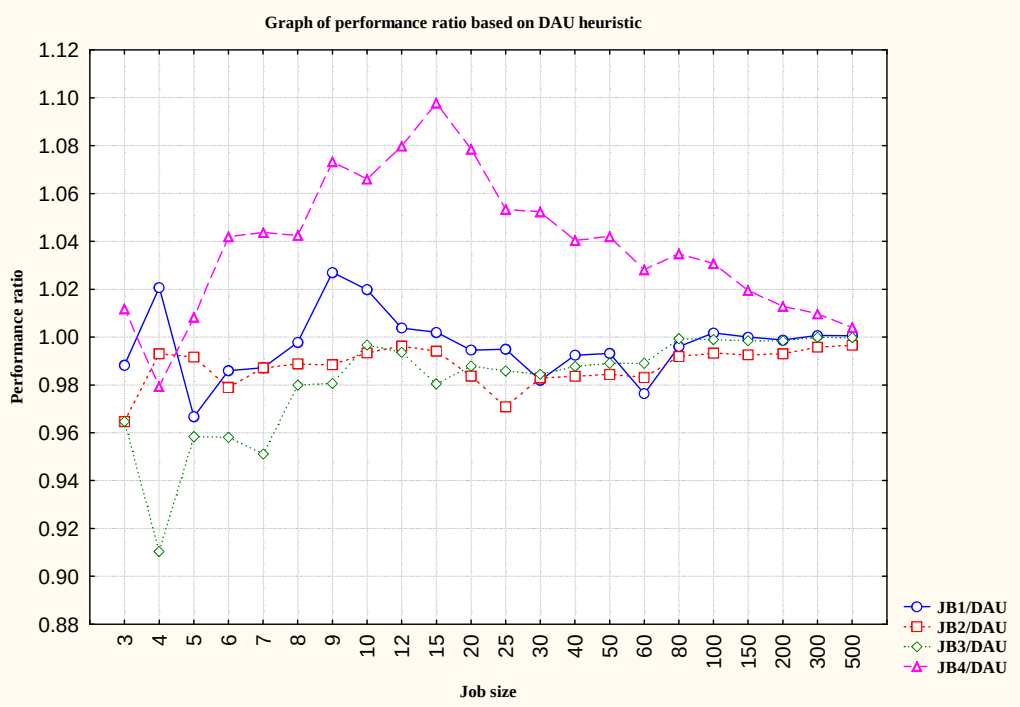

Fig. 1 Graph of performance ratio based on DAU heuristic

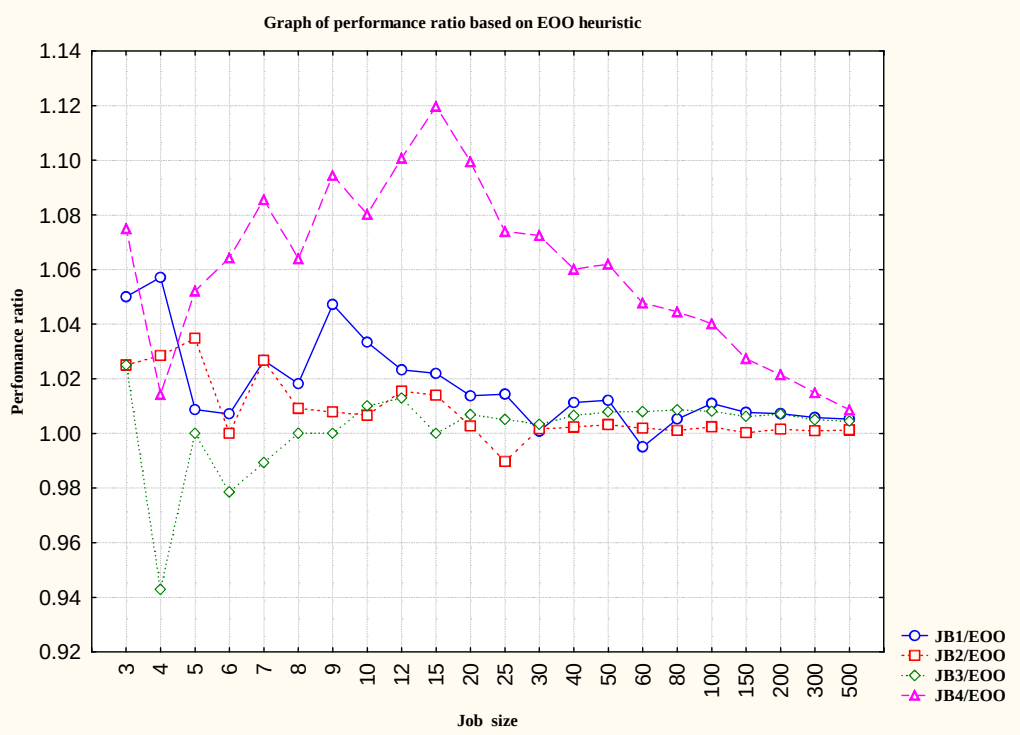

Fig. 2 Graph of performance ratio based on EOO heuristic

The heuristics are ranked such that, priority was given to them according to the percentage of times they obtain the lowest value of the number of tardy jobs for the fifty samples of each job size ($3 \leq n \leq 500$). The ranking was done with the alphabets

attached with percentage ranging from “a” to “f” in alphabetical order such that, the heuristic that ranks highest is given “a” and the one that follows it is given “b”. The order of performance observed with this test is EOO, JB2, JB3, JB1, DAU and lastly JB4 (Table 2).

TABLE 2 PERCENTAGE RANKING OF HEURISTICS THAT YIELDS LOWEST VALUE OF NUMBER OF TARDY JOBS

Job Size *n*	JB1 %	JB2 %	JB3%	JB4 %	EOO %	DAU %
3	96[c]	98[b]	98[b]	94[d]	100[a]	94[d]
4	90[c]	92[b]	100[a]	86[d]	92[b]	88[d]
5	98[b]	92[c]	100[a]	88[e]	100[a]	90[d]
6	92[c]	94[b]	100[a]	76[e]	94[b]	88[d]
7	82[d]	84[c]	96[a]	74[f]	92[b]	78[e]
8	86[d]	90[b]	94[a]	74[e]	94[a]	86[d]
9	76[d]	94[b]	98[a]	54[e]	98[a]	90[c]
10	76[e]	90[b]	88[c]	52[f]	94[a]	86[d]
12	76[e]	82[c]	86[b]	42[f]	94[a]	78[d]
15	74[d]	82[b]	96[a]	18[e]	96[a]	78[c]
20	72[d]	88[b]	82[c]	12[f]	92[a]	62[e]
25	62[d]	94[a]	84[c]	6[f]	92[b]	54[e]
30	82[c]	80[b]	78[d]	6[f]	86[a]	52[e]
40	56[d]	86[b]	74[c]	0[f]	90[a]	48[e]
50	44[d]	80[b]	74[c]	0[e]	92[a]	44[d]
60	80[b]	78[c]	64[d]	0[f]	90[a]	38[e]
80	40[d]	90[b]	64[c]	0[f]	90[a]	36[e]
100	18[e]	72[b]	30[c]	0[f]	90[a]	22[d]
150	10[d]	82[b]	30[c]	0[e]	94[a]	10[d]
200	0[d]	64[b]	10[c]	0[d]	92[a]	0[d]
300	0[c]	58[b]	0[c]	0[c]	74[a]	0[c]
500	0[c]	48[b]	0[c]	0[c]	94[a]	0[c]

The execution time (seconds) taken by all the heuristics under the various job sizes considered are shown in Table 3. The differences observed in the mean value of execution time taken by all the heuristics are not significant at 95% level (Table 3). Based on the mean value of execution time taken, the heuristics rank as follows: EOO, JB4, DAU, JB2 and lastly JB3 (Table 4)

TABLE 3 MEAN OF TIME OF EXECUTION OF JOBS

Job Size *n*	JB1(sec)	JB2(sec)	JB3(sec)	JB4(sec)	EOO(sec)	DAU(sec)
3	0.003	0.003	0.003	0.003	0.003	0.003
4	0.004	0.004	0.004	0.004	0.004	0.004
5	0.005	0.005	0.005	0.005	0.005	0.005
6	0.007	0.007	0.006	0.006	0.006	0.006
7	0.009	0.008	0.008	0.008	0.007	0.008
8	0.009	0.009	0.009	0.009	0.008	0.0091
9	0.01	0.01	0.01	0.01	0.009	0.001
10	0.013	0.012	0.012	0.012	0.011	0.012
12	0.016	0.015	0.014	0.014	0.013	0.014
15	0.018	0.02	0.02	0.019	0.017	0.019
20	0.031	0.028	0.027	0.025	0.023	0.025
25	0.04	0.036	0.034	0.032	0.031	0.033
30	0.045	0.046	0.041	0.041	0.04	0.043
40	0.066	0.066	0.067	0.065	0.06	0.061
50	0.094	0.09	0.089	0.072	0.083	0.089
60	0.122	0.115	0.103	0.091	0.112	0.119
80	0.242	0.18	0.147	0.125	0.144	0.151
100	0.273	0.235	0.235	0.198	0.199	0.208
150	0.525	0.527	0.551	0.43	0.495	0.507
200	1.198	1.005	1.013	0.842	0.945	1.051
300	3.129	2.818	2.711	2.472	2.671	2.752
500	11.391	10.225	10.614	10.875	10.417	11.753

TABLE 4 T-TEST OF THE PROPOSED HEURISTICS (JB1, JB2, JB3 AND JB4) IN RESPECT TO EOO AND DAU HEURISTICS

	T-test for Independent Samples Note: Variables were treated as independent samples									
Sample1 vs.Sample2	Mean Sample1	Mean Sample2	t-value	df	p	Number	Std.Dev. Sample1	Std.Dev. Sample2	F-ratio Variances	p Variances
EOO vs. JB1	0.695591	0.784091	-0.124319	42	0.901655	22	2.249115	2.467866	1.203982	0.674476
EOO vs. JB2	0.695591	0.702909	-0.010874	42	0.991376	22	2.249115	2.214974	1.031065	0.944779
EOO vs. JB3	0.695591	0.714682	-0.027892	42	0.977880	22	2.249115	2.290808	1.037419	0.933713
EOO vs. JB4	0.695591	0.698091	-0.003616	42	0.997132	22	2.249115	2.336317	1.079046	0.863254
DAU vs. JB1	0.766959	0.784091	-0.022744	42	0.981962	22	2.528161	2.467866	1.049461	0.912963
DAU vs. JB2	0.766959	0.702909	0.089379	42	0.929206	22	2.528161	2.214974	1.302783	0.549824
DAU vs. JB3	0.766959	0.714682	0.071872	42	0.943045	22	2.528161	2.290808	1.217957	0.655537
DAU vs. JB4	0.766959	0.698091	0.093836	42	0.925685	22	2.528161	2.336317	1.170971	0.720957

VIII. CONCLUSION

The scheduling problem of minimizing the number of tardy jobs with release dates on a single machine has been explored in this paper. In view of the NP-Hard nature of the problem, four heuristics (JB1, JB2, JB3, and JB4) were proposed for solving the problem. The proposed heuristics were compared with two heuristics (EOO and DAU) selected from the literature. All the heuristics (proposed and selected) were evaluated with respect to effectiveness (a measure of the closeness of value of number of tardy jobs to the optimal) and efficiency (a measure of execution speed or how fast solution can be obtained).

Based on performance with respect to both effectiveness and efficiency, the EOO heuristic is recommended for solving the single machine scheduling problems of minimizing the number of tardy jobs with release dates for problems involving more than 9 jobs. However, when the number of jobs is less than 10, the JB3 (one of the proposed heuristics) is recommended for solving the single machine scheduling problems of minimizing the number of tardy jobs with release dates.

REFERENCES

[1] Baptiste P. 1999a. An $O n^4$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. Operations Research Letters, vol. 24, pp. 175-180.

[2] Baptiste P., Peridy L. and Pinson E. 2003. A branch and bound to minimize the number of late jobs on a single machine with release time constraints. European Journal of Operation Research, vol. 144, pp. 1-11.

[3] Briand C., Ourari S. and Bouzouia B. 2006. On minimization of late jobs in single machine scheduling problem. Laboratoire d'Analyse et d'Architecture des Systemes, LAAS.

[4] Carlier, J. 1982. The one-machine sequencing problem. European Journal of Operational Research, vol. 111, pp. 42-47.

[5] Chrobak, M., Durr C., Jawor W., Kowalik L., and Kurowski M. (2006). A note on scheduling equal-length jobs to maximize throughput. Journal of Scheduling, vol. 9, pp. 71-73.

[6] Dauzere-Perez, S. 1995. Minimizing late jobs in the general, one-machine scheduling problem, European Journal of Operational Research, vol. 811, pp. 134-142.

[7] Dauzere-Perez, S. and Sevaux M. 1998. An efficient formulation for minimizing the number of the late jobs in the single machine scheduling. Technical 98/9/ Auto, Ecole des Mines de Nantes.

[8] Dauzere-Perez, S. and Sevaux M. 1999. Using Lagrangean Relaxation to Minimize the Weighted Number of Late Jobs on a Single Machine.

[9] Dauzere-Perez, S. and Sevaux M. 2004. An exact method to minimize the number of the late jobs in the single machine scheduling, Journal of Scheduling, vol. 7, pp. 405-420.

[10] French, S. 1982. Sequencing and Scheduling. Ellis-Horwood Limited.

[11] Lasserre, J. B. and Queyranne, M. 1992: Generic scheduling polhedra and anew mixed integer formulation for single machine scheduling. In 2nd Conference on Integer Programming and Combinatorial Optimization. Pittsburgh USA: Carnegie Mellon University, pp. 139-149.

[12] Lenstra J.K., Rinnooy Han A.H.G and Brucker, P. 1977. Complexity of machine scheduling problems, Annals of Discrete Mathematics, vol. 1, pp. 343-362.

[13] Lawler, E.L. 1990. A dynamic programming algorithm for the preemptive scheduling of a single machine to minimize the number of late jobs, Annals of Operations Research, vol. 26, pp. 125-133.

[14] Kise H., Ibaraki T. and Mine H. 1978. A solvable case of the one-machine scheduling problem with ready and due times. Operations Research. vol. 26, pp. 121-126.

[15] M'Hallah R. and Bulfin R. L. 2007. Minimizing the weighted number of tardy jobs on a single machine with release dates. European Journal of Operational Research, vol. 176, 727-744.

[16] Moore, J.M. 1968. A job, one machine sequencing algorithm for minimizing the number of late jobs, Management Science, vol. 15, pp. 102-109.

[17] Mundt A. and Wich T. 2007. Single machine scheduling with tardiness involved objective, Masters-Thesis, Linkopings Universitet, Department of mathematics.

[18] Oyetunji E.O. and Oluleye A.E. 2008. Heuristics for minimizing the number of tardy jobs on a single machine with release time. South African Journal of Industrial Engineering, vol. 192, pp. 183-196.

[19] Strusevich V.A. and Chris N.P. 2009. Fifty years of scheduling: A survey of milestones.

Biography

A.O. Akinrinde has just been admitted as a doctoral student in the Department of Electrical, Electronic and Computer Engineering, University of Kwazulu-Natal, South-Africa. He graduated with a BSc (Hons) in Electrical and Computer Engineering from Federal University of Technology, Minna, Nigeria. He has an MSc in Industrial Engineering from the University of Ibadan, Ibadan, Nigeria. His research interest includes production scheduling/operations management (optimization), and high voltage engineering.

E.O. Oyetunji is currently a Professor of Industrial and Production Engineering in the Department of Mechanical Engineering, Faculty of Engineering, Lagos State University, Epe Campus, Lagos, Nigeria. He graduated with a BSc (Hons) in Electrical Engineering from the University of Ilorin, Ilorin, Nigeria. He has an MSc and PhD in Industrial Engineering from the University of Ibadan, Ibadan, Nigeria. He has attended numerous national and international conferences. His research interest includes production scheduling/operations management (optimization), and algorithm design amongst others. He has published extensively in prestigious local and international journals. He is a Registered Engineer (COREN) and a Member of the following professional bodies: Nigeria Society of Engineers (NSE), Nigeria Institute of Industrial Engineering (NIIE), South African Institute of Industrial Engineering (SAIIE).

A.E. Oluleye is a Professor of Industrial and Production Engineering, Faculty of Engineering, University of Ibadan, Nigeria. He graduated with a BSc (Hons) in Agricultural Engineering from the University of Ibadan, Nigeria. He has an MSc in Agricultural Machinery Engineering from the Cranfield Institute of Technology, England. He has PhD in Industrial Engineering from the University of Ibadan, Nigeria. He has attended numerous national and international conferences. His research interest is in industrial engineering. He has published extensively in prestigious local and international journals. He is a Registered Engineer (COREN) and a Member of the following professional bodies: Nigeria Society of Engineers (NSE), Nigeria Institute of Industrial Engineering (NIIE), American Institute of Industrial Engineering (AIIE), Nigeria Society of Agricultural Engineers (NSAE), and New York Academy of Science.