Journal of the Nigerian Mathematical Society
Vol. 33, pp. 321-331, 2014

A NOTE ON JUST-IN-TIME SCHEDULING ON FLOW SHOP MACHINES

M. O. ADAMU[1], N. BUDLENDER AND G. A. IDOWU

ABSTRACT. In this paper, the scheduling to maximize the weighted number of Just-In-Time jobs is considered. This problem is known to be NP Complete for when the due date is at a point in time indicating no efficient optimal solution is feasible in reliable time. Due dates with interval in time are considered in this work. The problem formulation is suggested, two greedy heuristics are proposed for solving the problem. A numerical example to illustrate its use and extensive computational experiments performed with promising results are presented. Likely areas of extensions are provided.

Keywords and phrases: Just-In-Time, NP Complete, Flow Shop, Machines, Scheduling

1. INTRODUCTION

Just-In-Time(JIT) production scheduling system is an evolving area in scheduling. JIT is a production strategy that strives to improve return on investment by reducing in-process inventory and associated carrying costs. In the last two decades, several research works have dealt with only single and parallel machines environments with very few considering JIT scheduling in the flow shop machine environment.

In this paper, we consider scheduling n jobs on m machines to maximize the weighted number of JIT jobs on a flow shop environment. An m-machine flow shop problem consists of n independent jobs on m machines simultaneously available from time zero. Let each job have an interval rather than a point in time, called due window of the job. The left and right ends of the window are the earliest start time, $a_j \geq 0$ (i.e. instant at which a job becomes available) and the latest due date, $d_j \geq 0$ (instant by which processing or delivery of a job must be completed). There is no penalty when a job is completed within the job due window, but earliness (tardiness) penalty is incurred when a job is completed before the

Received by the editors October 11 2013; Revised: January 31, 2014; Accepted: February 5, 2014

[1]Corresponding author

job earliest start time (after the job latest due date). When a sequence of jobs is determined, the jobs undergo through operation on all machines without changing their sequence. Each job is done at most once on each machine. Each machine can only process one job at a time. No job can be preempted. The relevance of our problem in Production/Industrial systems cannot be over emphasized. Our problem is a Just-In-Time problem (JIT) where jobs must be ready at specific times in order to meet some important situations. These jobs must have to go through several machines before they are ready. Production of perishable/imperishable items (e.g. drugs, bulbs, vehicles, refrigerators, food, e.t.c) would require them to go through several production processes before they are ready for use.

Another application is in production units with no capacity to allow inventories where the due windows are determined by the pick-ups are made by customers. The goal would be to as much as possible meet the set time by customers so as not to incur penalty of loss of contract, product waste (perishable) if due date is missed.

Using the problem classification of Graham *et al.*[7], our problem is $Fm \mid\mid \sum w_j(U_j + V_j)$. That is, minimizing the weighted number of early and tardy jobs on m Flow Shop machines. However, the dual of this problem is maximizing the weighted number of on-time jobs (JIT) on m Flow Shop. The remaining parts of the paper are as follows: Section two considers the literature review. The problem formulation is outlined in section three. The proposed algorithms for our problem are presented in section four. In section five, the problem generation and computational results are enumerated and finally, in section six, the conclusion.

2. LITERATURE REVIEW

Bulfin and M'Hallah [5] constructed an exact algorithm to solve the weighted number of tardy jobs on two-machine flow shop scheduling problem ($F2||\sum w_jU_j$). They provided a branch and bound algorithm that used surrogate relaxation resulting in a multiple-choice knapsack providing bounds. Extensive computational experiments conducted indicate problems with 100 jobs can be solved quickly. This problem is NP-Hard in the strong sense. Some others that considered not too different problems are Lenstra [14], Gupta and Hariri [9], Jozefowska *et al.* [13] and Ho and Gupta [11].

Scheduling to maximize the weighted number of Just-In-Time jobs on the Flow Shop machine that should be completed exactly on their due dates was considered by Choi and Yoon [6]. They

proved that this problem is NP-Complete. When the weights are all identical, they showed that the problem can be solved in polynomial time. For $m \geq 3$ with identical jobs weights is NP-hard in the strong sense.

Yeung *et al.* [16] addressed the two due window scheduling problems to minimize the weighted number of early and tardy jobs in a two-machine Flow Shop, where the window size is externally determined. They introduced dominance properties and theorems, lower bounds and upper bounds on the window location. They showed the problems are NP-Hard in the ordinary sense. They proposed a pseudo polynomial dynamic programming algorithms for the problems, $F||\sum(u_jU_j + v_jV_j)$ and $F||\sum(u_jU_j + v_jV_j) + L(d)$. Li *et al.* [15] considered our problem on the single machine. Others that considered it on parallel machines are Adamu and Abass [1], Janiak *et al.* [12], Adamu and Adewumi [2]. A comprehensive reviews can be found in Adamu and Adewumi[3][4]. The problem considered is NP complete as shown by Choi and Yoon [6] and finding an optimal solution is unlikely, hence, the need to find approximate solution to the problem.

3. PROBLEM FORMULATION

The mathematical formulation for the problem, $Fm||\sum w_j(U_j + V_j)$ is discussed in this section. For ease of presentation, we will take the dual, which is to maximize the JIT jobs, $Fm||\sum w_j x_{ijk}$.

Let x_{ijk} be one if job j is in position k on machine i and on-time and zero otherwise, p_{ij} be the processing time of job j on machine i, w_j be the weight of job j, t_{ik} be the start time on the ith machine in the kth position, a_j be the earliest due date of job j, d_j be the latest due date of job j. The objective function is the maximization of the weights of JIT jobs. Constraint (1) ensures each job will have m positions on the m machines. Constraint (2) specifies that each position on each machine cannot be occupied by more than one job. In constraint (3), jobs are prevented from finishing before their earliest due date. Similarly, in constraint (4), no job scheduled in position k has its completion time greater than its latest due date if it is on-time. Constraint (5) ensures that the job j sequenced in position k on machine i will not start before the completion time of the job j on machine $i-1$ in position k. In constraint (6), the start time in position k on any machine i is greater or equal to zero. Finally, in constraint (7), a binary variable $x_{ijk} = 1$ if job j is scheduled on-time at position k on machine i.

Max:

$$Z = \sum_{i=1}^{m} \sum_{j=1}^{n} w_j x_{ijk}$$

subject to:

$$\sum_{j=1}^{m} \sum_{k=1}^{n} x_{ijk} = m \qquad \forall j = 1, 2, \ldots, n \tag{1}$$

$$\sum_{j=1}^{n} x_{ijk} = 1 \qquad \forall i = 1, 2, \ldots, m; \qquad \forall k = 1, 2, \ldots, n \tag{2}$$

$$t_{1k} - \sum_{j=1}^{n} \max\{t_{1,k-1}, a_j - \sum_{i=1}^{m} p_{ij}\} x_{ijk} \geq 0 \qquad \forall k = 1, 2, \ldots, n \tag{3}$$

$$t_{i-1,k} + \sum_{j=1}^{n} (p_{ij} - d_j) x_{ijk} \leq 0 \qquad \forall i = 2, \ldots, m; \qquad \forall k = 1, 2, \ldots, n \tag{4}$$

$$t_{i-1,k} + \sum_{j=1}^{n} p_{ij} x_{ijk} \leq t_{ik} \qquad \forall i = 2, \ldots, m; \qquad \forall k = 1, 2, \ldots, n \tag{5}$$

$$t_{ij} \geq 0 \qquad \forall i = 1, 2, \ldots, m; \qquad \forall k = 1, 2, \ldots, n \tag{6}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i = 1, 2, \ldots, m; \qquad \forall j = 1, 2, \ldots, n; \qquad \forall k = 1, 2, \ldots, n \tag{7}$$

4. ALGORITHM

Two greedy heuristics are proposed for the solution to the problem $Fm||\sum w_j(U_j + V_j)$ which are $F1$ and $F2$.

4.1 ALGORITHM F1

(1) Re-index the jobs $\ni a_1 \leq a_2 \leq \ldots \leq a_n$
(2) $T := \emptyset; L := \emptyset; Q := \{J_1, J_2, \ldots, J_n\}; t_{10} := 0; i := 1, 2, \ldots, n; |T| := 0$
(3) Assign J_1 to machines and break tie by highest $\frac{w_j}{p_{ij}}$
(4) For $j := 1$ to n do
    For $i := 2$ to m do
        $t_{1j} = \max\{t_{1,j-1}, a_j - \sum_{i=1}^{m} p_{ij}\} + p_{ij}$
        $t_{ij} = t_{i-1,j} + p_{ij}$
    End for
    If $t_{mj} \leq d_j$ then
      $T := T \cup \{J_j\}$; $Q := Q \setminus \{J_j\}$; $j := j + 1$; $|T| := |T| + 1$
      Else step 5
    End if
  End for
(5) Find jobs J_r in T with $w_r < w_j$
  For $l := 1$ to $|J_r|$ (where $w_1 < w_2 < \ldots < w_r$)

Remove Job J_l from T ($T := T \setminus \{J_l\}$)
Reassign Job J_j
If $t_{mj} \leq d_j$ then
$T := T \cup \{J_j\}; L := L \setminus \{J_l\}; j := j+1; |T| := |T|+1$
Else
Step 4
End if
End for
$L := L \setminus \{J_j\}; j := j+1$
Step 4

(6) Stop (Find total weights in L or T)

4.2 ALGORITHM F2

(1) Re-index the jobs $\ni a_1 \leq a_2 \leq \ldots \leq a_n$
(2) $T := \emptyset$; $L := \emptyset$; $Q := \{J_1, J_2, \ldots, J_n\}$; $t_1 0 := 0$; $i := 1, 2, \ldots n$; $|T| := 0$
(3) Assign J_1 to machines and break tie by highest $\frac{w_j}{p_{ij}}$
(4) For $j := 1$ to n do
For $i := 2$ to m do
$t_{1j} = \max\{t_{1,j-1}, a_j - \sum_{i=1}^{m} p_{ij}\} + p_{ij}$
$t_{ij} = t_{i-1,j} + p_{ij}$
End for
If $t_{mj} \leq d_j$ then
$T := T \cup \{J_j\}$; $Q := Q \setminus \{J_j\}$; $j := j+1$; $|T| := |T|+1$
Else
step 5
End if
End for
(5) Find jobs J_r in T with $\bar{w_r} = \frac{w_r}{\{\frac{\sum_{i=1}^{m} p_{ir}}{m}\}} < \bar{w_j} = \frac{w_j}{\{\frac{\sum_{i=1}^{m} p_{ij}}{m}\}}$
For $l := 1$ to $|J_r|$ (where $\bar{w_1} \leq \bar{w_2} \leq \ldots \leq \bar{w_r}$)
Remove Job J_l from T ($T := T \setminus \{J_l\}$)
Reassign Job J_j
If $t_{mj} \leq d_j$ then
$T := T \cup \{J_j\}$;$L := L \setminus \{J_l\}$; $j := j+1$; $|T| := |T|+1$
Step 4
Else
End if
End for
$L := L \setminus \{J_j\}$; $j := j+1$
Step 4

(6) Stop (Find total weights in L or T)

The time complexity of these algorithms is at most $O(n^2)$. L contains jobs that are early or tardy while T contains the on-time jobs

NUMERICAL EXAMPLE

A numerical example is presented below in Table 1 to show case the proposed algorithms and results derived from them in Figures 1 and 2.

Table 1. Numerical Example for $F_m||w_j(U_j + V_j)$

J_j	a_j	d_j	Processing Time (M_1)	Processing Time (M_2)	Processing Time (M_3)	Weight(w_j)	$\frac{w_j}{[\frac{(p_1+p_2+p_3)}{3}]}$
1	0	6	2	1	2	2	1.2
2	2	9	3	2	1	4	2
3	4	10	2	2	2	6	3
4	8	16	3	2	2	5	2.143
5	9	15	1	1	1	1	1.0
6	11	19	4	2	2	2	0.75
7	13	19	3	3	1	7	3

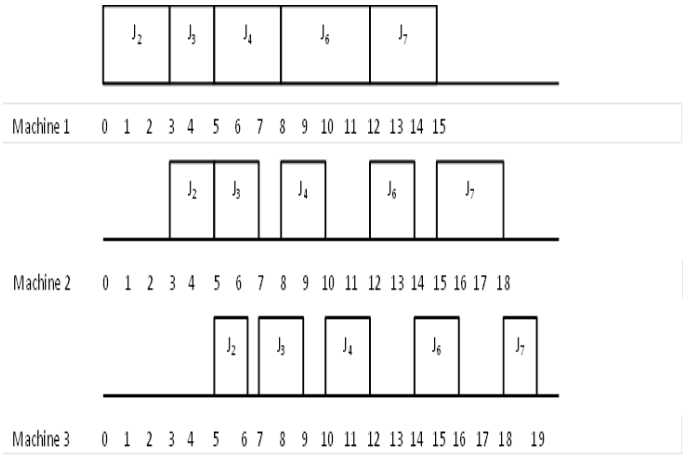

Fig. 1. Gantt chart diagram for Algorithm F1.

Following the numerical example, the number of tardy jobs is 2 (J_1 and J_5) and tardy weight is 3 and on-time weight is 24 for Algorithm F1. For Algorithm F2, the number of tardy jobs is 2 (J_1 and J_6) and tardy weight is 4 and on-time weight is 23.

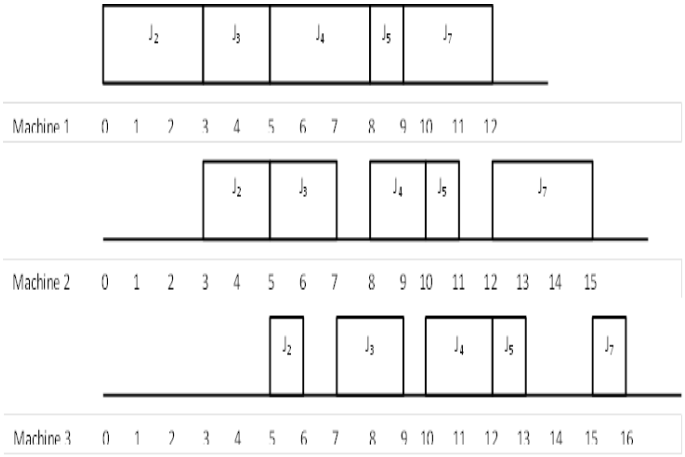

Fig. 2. Gantt chart diagram for Algorithm F2.

5. PROBLEM GENERATION AND COMPUTATIONAL RESULTS

5.1 PROBLEM GENERATION

The heuristics F1 and F2 were tested on problems generated with $n = 100, 200, 300, 400$ and 500 and the number of machines, m, set at levels $m = 2, 3, 5, 8, 10$. Using similar problem instance as Bulfin and M'Hallah [5], Hariri and Potts [10] and Gupta and Hariri [9], two parameters $k1$ and $k2$ were chosen to provide upper and lower bounds for the due dates. $k1 = \{0.2, 0.4, 0.6, 0.8\}$ and $k2 = \{0.4, 0.6, 0.8, 1.0\}$, where $k1 < k2$.

Let $P = (\frac{\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij} + \sum_{j=1}^{n}(n-1)p_{i^\star j}}{n})$ be an estimate of the maximum completion time P, obtained by identifying the machine $i^\star$ with the largest total processing time. The integer earliest due date, a_j, was randomly generated from the uniform distribution $[0, Pk1]$ and the integer latest due date, d_j, was randomly generated from the uniform distribution $[a_j + \sum_{i=1}^{m} p_{ij}, a_j + \sum_{i=1}^{m} p_{ij} + Pk2]$. The integer weights, w_j, was randomly generated from the interval $[1, 10]$. The integer processing times, p_{ij}, were randomly generated from the uniform distribution [1,99]. For each of the ten pairs of k1 and k2 parameters, ten instances were randomly generated for $n = 100$, 20 instances were randomly generated for $n = 200$, e.t.c. For each n and m, 50 replications were generated and the average of the tardy weights and time are tabulated.

Table 2. Computational result for Algorithms F1 and F2.

n	m		2	3	5	8	10
100	wT	F1	210	165	129	102	88
		F2	208	162	129	102	88
	Time	F1	0.09008	0.07204	0.06548	0.05454	0.0335
		F2	0.09458	0.07362	0.07414	0.04656	0.03642
200	wT	F1	430	357	258	204	174
		F2	427	354	255	201	174
	Time	F1	0.28582	0.19344	0.17752	0.1273	0.1092
		F2	0.39988	0.27576	0.23342	0.16904	0.15506
300	wT	F1	654	528	405	282	242
		F2	651	525	402	282	242
	Time	F1	0.82898	0.45438	0.30586	0.25558	0.30678
		F2	1.15564	0.66986	0.49534	0.39692	0.46932
400	wT	F1	868	705	522	378	324
		F2	864	702	519	378	323
	Time	F1	1.41616	0.7613	0.51796	0.62024	0.47296
		F2	2.06166	1.2648	0.94534	0.83614	0.86924
500	wT	F1	1088	888	642	471	400
		F2	1082	885	639	471	400
	Time	F1	2.33094	1.41744	0.79594	0.52074	0.5847
		F2	3.85076	2.48918	1.69752	1.00774	1.21202

5.2 COMPUTATIONAL RESULTS

The computational results and graphs for the time performance of the two algorithms are presented in Table 2, Figures 3 and 4. The first two cells represent the tardy weights for Algorithms F1 and F2. The time in seconds are given below in the next two cells for the algorithms. In all cases considered, it could be ascertained that Algorithm F2 out performed F1. Conversely, the Algorithm F1 used lesser computational time compared with Algorithm F2. It was observed that as the number, n, increases the gap widens further between the computational times of both algorithms. The reason for this could be as a result of step 5 of Algorithm F2 where the number of arithmetic operations is a factor. The computational times of the algorithms are proportional to the number of jobs. That is, as the number of jobs increases, the computational times mount. For fixed number of jobs, the problem takes lesser computation time as the number of machines increases.

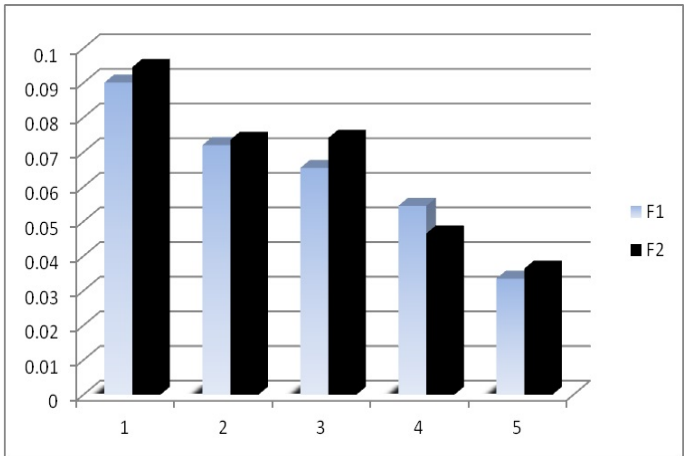

Fig. 3. Time performance of F1 and F2 when n = 100

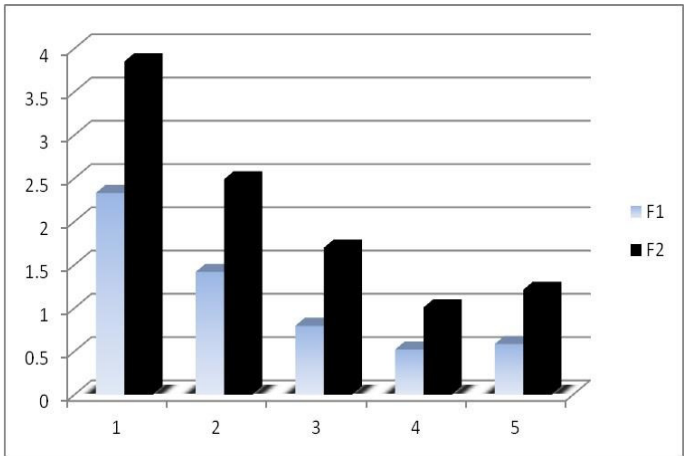

Fig. 4. Time performance of F1 and F2 when n = 500.

6. CONCLUSION

In this work, scheduling to maximize the weighted number of JIT jobs on m Flow Shop machines was considered. The dual of this problem is also known as minimizing the weighted number of early and tardy jobs on m Flow Shop machines. The problem formulation for the problem is given and two greedy heuristics presented. Numerical example and computational experiments were performed with results showing the effectiveness of the heuristics. Further research should seek to improve on these results by using Metaheuristics methods, find exact solutions for small samples where

possible. In addition, approximation and pseudo-polynomial algorithms could be developed.

REFERENCES

[1] M.O. Adamu and O. Abass, *Parallel Machine Scheduling to Maximize the Weighted Number of Just-In-Time Jobs.*, Journal of Applied Science and Technology, **15** (1,2) 27-34, 2010.

[2] M.O. Adamu and A.O. Adewumi, *Metaheuristics for Scheduling on Parallel Machines to minimize the Weighted Number of Early and Tardy Jobs*, International Journal of Physical Sciences, **7** (10) ,1641-1652, 2012.

[3] M.O. Adamu and A. Adewunmi, *Minimizing the Weighted Number of Tardy Jobs on Multiple Machines: A Review.*Submitted. 2013.

[4] M.O.Adamu and A. Adewunmi, *A Survey of Single machine Scheduling to Minimize Weighted Number of Tardy Jobs.* Journal of Industrial and Management Optimization, **10**(1), 219-241, 2014.

[5] R.L. Bulfin and R. M'Hallah,*Minimizing the Weighted Number of Tardy Jobs on a Two-Machine Flow Shop,* Computers and Operational Research, **30**, 1887-1900, 2003.

[6] B.C. Choi, and S.H. Yoon, *Maximizing the Weighted Number of Just-In-Time Jobs in Flowshop Scheduling.*Journal of Scheduling, **10**, 237-243, 2007.

[7] R.L. Graham, E.L. Lawler, T.K. Lenstra, and A.H.G. Rinnooy Kan, *Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey.* Annals of Discrete Mathematics. **5**, 287-326, 1979.

[8] J.N.D. Gupta, and A.M.A. Hariri,*Integrating Job Selection and Scheduling in a Flowshop.*Working Paper, Department of Management, Ball State University, Muncie, IN, 1994.

[9] J.N.D.Gupta and A.M.A. Hariri ,*Two Machine Flow Shop to Minimize Number of Tardy Jobs.* Journal of Operational Research Society, **48**, 212-220, 1997.

[10] A.M.A. Hariri and C.N. Potts,*A Branch and Bound Algorithm to Minimize Number of Late Jobs in a Permutation Flow Shop.* European Journal of Operational Research, **38**, 228-237, 1989.

[11] J.C. Ho, and J.N.D. Gupta, *Flowshop Scheduling with Dominant machine.* Computers and Operations Research, **22**, 237-246, 1995.

[12] A. Janiak, W.A. Janiak, and R. Januszkiewicz, *Algorithms for Parallel Processor Scheduling with Distinct Due Windows and Unit-Time Jobs* Bulletin of the Polish Academy of Sciences Technical Sciences, **57**, (3), 209-215, 2009.

[13] J. Josefowska, B. Jurisch, and W. Kubiak, *Scheduling Shops to Minimize the Weighted Number of Late Jobs.* Operations Research Letters, **10**, 27-33, 1994.

[14] J.K.Lenstra, A.G.H. Rinnooy, and P. Brucker,*Complexity of Machine Scheduling Problems.* Annals of Discrete Mathematics, 1, 343-362, 1977.

[15] Li, C.L., Cheng, T.C.E. and Z.L. Chen,*Single Machine Scheduling to Minimize the Weighted Number of Early and Tardy Agreeable Jobs.* Computers and Operations Research, **22**(2), 205-219, 1995.

[16] W.K. Yeung, C. Oğuz and T.C.E. Cheng,*Two-Machine FlowShop Scheduling with Common Due Window to Minimize Weighted Number of Early and Tardy Jobs.* Naval Research Logistics, **5**, 593-599, 2009.

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF LAGOS, AKOKA, NIGERIA

E-mail address: madamu@unilag.edu.ng

SCHOOL OF MATHEMATICS, STATISTICS AND COMPUTER SCIENCE, UNIVERSITY OF KWAZULU-NATAL, SOUTH AFRICA
E-mail address: nigel.budlender@gmail.com
DEPARTMENT OF COMPUTER SCIENCE, LAGOS STATE UNIVERSITY, LAGOS, NIGERIA
E-mail address: gbolahan.idowu@lasu.edu.ng