HEURISTIC APPROACH FOR SOLVING FLOW SHOP MACHINE SCHEDULING PROBLEMS

Adamu, M.O.[1]**, Idowu, G.**[2] **and Budlender, N.**[3]
[1]Department of Mathematics, University of Lagos
Akoka, Yaba, Lagos, Nigeria.
(madamu@unilag.edu.ng)
[2]Lagos State University,Ojo, Lagos, Nigeria.
(gbolahan.idowu@lasu.edu.ng)
[3]School of Mathematics, Statistics & Computer Science
University of Kwazulu-Natal, South Africa.
(nigel.budlender@gmail.com)

ABSTRACT
The problem of scheduling n jobs on m flow shop machines to maximize the (weighted) number of Just-In-Time jobs is considered. It is known that this problem is NP-Complete even for a single machine, indicating that no efficient optimal solution can be found in reliable time for even fairly large instance problems. In this research, two greedy heuristic solutions are proposed and compared with an optimum solution found by Xpress-MP using small problem instances. Computational results and analysis for various scales of instances show that the greedy heuristic algorithms performed creditably well when compared with an optimal solution using small problem instances. The quality and efficiency of the heuristics coupled with solutions to large instance problems are highlighted.

Keywords: Inventory, Heuristics, Combinatorial Optimization, Flow Shop, Scheduling

1. Introduction
In this paper, we consider scheduling n jobs on m machines to maximize the (weighted) number of Just-In-Time (JIT) jobs in a flow shop environment. An m-machine flow shop problem consists of n independent jobs on m machines simultaneously available from time zero. It is assumed 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 job earliest start time (after the job latest due date). When a sequence of jobs is determined, the jobs undergo 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 pre-empted.

The relevance of JIT problem in Production/Industrial systems cannot be over emphasized. The problems are Just-In-Time problems 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/non-perishable items as an application (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-up times and 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 dates are missed.

Using the problem classification of Graham *et al* (1979), our problems are $Fm||(U_j + V_j)$ and $Fm| |w_j(U_j + V_j)$. The F describes the shop (machine) environment for flow shop machines and m describes the number of machines. The space between the bars is for constraints on the jobs, which include the following: preemption, release time, setup, batching precedence, etc. In our case, none is considered. The symbols U_j and V_j are binary (0 or 1) and indicate whether a job is scheduled early or tardy respectively; 0 is used when the job is scheduled on-time and 1 is used if it is not (i.e. early or tardy). That is, the problems being considered are minimizing the number of early and tardy jobs and minimizing the (weighted) number of early and tardy jobs on *m* Flow Shop machines. However, the dual of these problems are maximizing the number of on-time jobs and maximizing the (weighted) number of on-time jobs (JIT) on *m* Flow Shop machines. The remaining parts of the paper are as follows: In Section two, the notations and definition of terms used are stated. Section three considers the literature review. The problem formulation is outlined in Section four. The heuristic algorithms for the problems are presented in Section five. In Section six, the problem generation and computational results are enumerated. The discussion of findings are presented in Section seven and finally, in Section eight, the conclusion is presented.

2. Notation and Definitions

In this section, terms and notation used in this paper are presented. Some of the terms with their definitions are:

Greedy Heuristics: Greedy heuristics represent a common subtype of construction heuristics. As the name suggests, the action of these heuristics is such that at any decision point the option corresponding to the greatest immediate 'return' is chosen.

NP: Nondeterministic time- when no deterministic polynomial-time algorithm is available to solve the problem.

NP-Hard: A problem is NP-hard if it is at least as hard as any problem in NP.

NP-Complete: A problem is NP- complete if it is both NP-hard and in NP. NP-complete problems are often addressed by using heuristic methods and approximation algorithms.

Just-In-Time (JIT) scheduling: is concerned with scheduling jobs to minimize the total cost associated with both early and tardy completion.
The various notations used throughout the paper are as follows:

$$x_{ijk} = \begin{cases} 0 \ if \ job \ j \ is \ ontime \ at \ position \ k \ on \ machine \ i \\ 1 \qquad\quad if \ otherwise \end{cases}$$

t_{ik}: is the start time on ith machine in the kth position,

p_{ij}: is the processing time of job j on machine i,

a_j: is the earliest due date of job j,

d_j: is the latest due date of job j,

w_j: is the weight of job j,

$\ni$: such that,

T, L and Q: are the set of on-time jobs, tardy (late) jobs and unscheduled jobs respectively,

3. Literature Review

During the past few decades, a considerable amount of work has been conducted on scheduling on multiple machines [Adamu and Adewumi (2016)] and single machine [Adamu and Adewumi (2014)] in order to minimize the number of tardy jobs. Adamu *et al* (2014) proposed two greedy heuristics for solving the weighted case of maximizing the number of JIT jobs on flow shop machines with a numerical example and some computational experiments. Yeung *et al* (2009) addressed 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. Yeung *et al* (2009) introduced dominance properties and theorems, lower bounds and upper bounds on the window location. Yeung *et al* (2009) showed the problems are NP-Hard in the ordinary sense. Yeung *et al* (2009) proposed a pseudo polynomial dynamic programming algorithms for the problems ($F||\sum(u_jU_j + v_jV_j)$ and $F||\sum(u_jU_j + v_jV_j) + L(d)$). Scheduling to maximize the weighted number of Just-In-Time jobs that should be completed exactly on their due dates was considered by Choi and Yoon (2007) in which 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 when the number of machines, $m \geq 3$ with identical job weights, the problem is NP-hard in the strong sense.

Hariri and Potts (1989) solved $F2||\sum U_j$ problem using a branch and bound and developed three lower bounds: The first was obtained by solving a single machine subproblem and the second was derived from improvement procedures considering all subproblems simultaneously. Some 25 job problems were solved in 60 seconds cut off time. Bulfin and M'Hallah (2003) constructed an exact algorithm to solve the weighted number of tardy jobs on two-machine flow shop scheduling problem ($F2||\sum w_jU_j$). Bulfin and M'Hallah (2003) 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. Lenstra *et al* (1977) had shown that the number of tardy jobs in a two-machine flow shop, even if all jobs have a common due date, ($F2|d_j = D|\sum U_j$) is NP-hard in the strong sense. The others that considered the same problem are Gupta and Hariri (1994) and Jozefowska *et al* (1994). Ho and Gupta (1995) proposed polynomial algorithms to optimally solve two special cases of the $F2|d_j|\sum U_j$ problem. For the problem of $F2|d_j|\sum U_j$, Gupta and Hariri (1997) constructed several heuristics and gave four polynomially solutions to some special cases for the problem. Croce *et al* (2000) presented some structural properties of the $F2|d_j = d|\sum U_j$ problem, i.e. minimizing the total number of tardy jobs against a common due date. A branch and bound was proposed to find an optimal solution to the problem. The algorithm was able to solve up to 900 jobs instance. Lin (2001) proved that some special cases previously known as NP hard were actually NP-hard in the strong sense. He further constructed an $O(n^2)$ algorithm to solve a special case when all jobs have a common processing time on the first machine and agreeable processing times and due dates on the other machines. Xiang *et al* (2000) proposed polynomial time algorithms for solvable

special cases of F|*idm*|$\sum U_j$ and F|*ddm*|$\sum U_j$, where *idm* is an increasing series of dominating machines and *ddm* a decreasing one.

Shabtay and Bensoussan (2012) provided a pseudo-polynomial time algorithm to solve the F2||$\sum w_j U_j$ and proved that it is NP-hard in the ordinary sense. They showed how the pseudo-polynomial algorithm can be converted to a fully polynomial time algorithm scheme (FPTAS). They further proved that the same problem is strongly NP-hard for both a two-machine job shop and open shop scheduling system. Baki and Vickson (2004) considered the one operator two machine flow shop problem to minimize the weighted number of tardy jobs and proved that the flow shop total tardy jobs problem is NP-hard. Baki and Vickson (2004) gave two pseudo-polynomial dynamic algorithms for this problem (F2, S1||$\sum w_j U_j$). Desprez *et al* (2006) considered the flow shop in which each operation needs several resources, some of these resources being polyvalent and used a Genetic Algorithm for solving the problem and evaluated the effectiveness of the proposed method against a commercial software package. Lawler and Moore (1969) addressed the F2|d_j = D|$\sum U_j$ problem and provided a pseudo-polynomial dynamic programming algorithm with time complexity $O(nD^2)$. Ucar and Tasgetiren (2006) considered the problem Fm||$\sum U_j$ and used discrete Particle Swarm Optimization (PSO) to determine the sequence of *n* jobs to be processed through *m* machines that minimize the number of tardy jobs. The PSO algorithm gave promising solutions by means of the proposed SPV (Smallest Position Value) heuristic rule.

In this paper, comparison is made with solution obtained from an existing Mixed Integer Linear Program solver for both weighted and unweighted cases of scheduling to maximize the Just-In-Time jobs on a Flow Shop machine. Extensions to large sample problems are also considered.

4. Problem Formulation

The mathematical formulation for the problems, Fm||$\sum(U_j+V_j)$ and 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||$\sum x_{ijk}$ and Fm||$\sum w_j x_{ijk}$.

The objective function given below is the maximization of the (weighted) number 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 the following constraints:

$$\sum_{i=1,m}\sum_{k=1,n} x_{ijk} = m \qquad \forall\, j = 1,2,\ldots,n \qquad (1)$$

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

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

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

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

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

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

5. Algorithms

Two greedy heuristics are proposed for the solution to the problems $Fm||\sum(U_j+V_j)$ and $Fm||w_j(U_j+V_j)$ which are denoted as Algorithm F1 and Algorithm F2. Each of the two algorithms has two variant, one for minimizing the number of early and tardy jobs and the second for minimizing the weighted number of early and tardy jobs. The differences can be found in steps 3 and 5.

Algorithm F1

1. Re-index the jobs $\ni a_1 \leq a_2 \leq \ldots \leq a_n$

2. T:=Ø; L:= Ø; Q:={$J_1, J_2, \ldots, J_n$}; t_{10}:=0; *i*:=1,2, . . . ,n; Q is the set of unscheduled jobs, T is the set of scheduled jobs and L is the set of tardy or early jobs. |T|:=0

3. Assign J_1 to machines and break tie by smallest p_{ij} or highest w_j/p_{ij}

4. For j: = 1 to n do
   For i:= 1 to m do
   $t_{1j} = \text{Max}\{t_{1j-1}, a_j - \sum_{i=1,m} p_{ij}\} + p_{ij}$
   $t_{ij} = t_{i-1j} + p_{ij}$
   End for
   If $t_{mj} \leq d_j$ then
   T:= T U$\{J_j\}$; Q:= Q\$\{J_j\}$; j:= j +1; |T|:= |T| + 1
   Else step 5
   End if
   End for

5. Find jobs J_r in T with $p_r > p_j$ or $w_r < w_j$
   For l:= 1 to $|J_r|$
   Remove Job J_l from T
   Reassign Job J_j

   If $t_{mj} \leq d_j$ then
   T:= T U$\{J_j\}$; L:= L U $\{J_l\}$; T:= T\$\{J_l\}$; j:= j +1; |T|:= |T| + 1

   Step 4
   Else
   End if
   End for
   L:= L U $\{J_j\}$; j:= j +1
   Step 4

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

In Step 3, we break tie by smallest p_{ij} for minimizing the number of early and tardy jobs and highest w_j/p_{ij} for the weighted number of early and tardy jobs. In

Step 5, $p_r > p_j$ is used for minimizing the number of early and tardy jobs and $w_r < w_j$ for the weighted number of early and tardy jobs. L contains the weights of early and tardy jobs while T contains the JIT jobs.

Algorithm F2

1. Re-index the jobs $\ni a_1 \leq a_2 \leq \ldots \leq a_n$
2. T:=Ø; L:= Ø; Q:={$J_1, J_2, \ldots, J_n$}; t_{10}:=0; i:=1,2, . . . ,n; |T|:=0
3. Assign J_1 to machines and break tie by smallest p_{ij} or highest w_j/p_{ij}

4. For j:= 1 to n do
   For i:= 1 to m do
   $t_{1j} = \text{Max}\{t_{1j-1}, a_j - \sum_{i=1,m} p_{ij}\} + p_{ij}$
   $t_{ij} = t_{i-1j} + p_{ij}$
   End for
   If $t_{mj} \leq d_j$ then
   T:= T U{J_j}; Q:= Q\{J_j}; j:= j +1; |T|:= |T| + 1
   Else step 5
   End if
   End for

5. Find jobs J_r in T with
   $\bar{p}_r = \frac{\sum_{i=1,m} p_{ir}}{m} > \bar{p}_j = \frac{\sum_{i=1,m} p_{ij}}{m}$ or $\bar{w}_r = \frac{w_r}{\left\{\frac{\sum_{i=1,m} p_{ir}}{m}\right\}} < \bar{w}_j = \frac{w_j}{\left\{\frac{\sum_{i=1,m} p_{ij}}{m}\right\}}$
   For l:= 1 to $|J_r|$
   Remove Job J_l from T
   Reassign Job J_j
   If $t_{mj} \leq d_j$ then
   T:= T U{J_j}; L:= L U {J_l}; T:= T\{J_l}; j:= j +1; |T|:= |T| + 1
   Step 4
   Else
   End if
   End for
   L:= L U {J_j}; j:= j +1
   Step 4

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

In Step 3, we break tie by smallest p_{ij} for minimizing the number of early and tardy jobs and highest w_j/p_{ij} for the weighted number of early and tardy jobs. In Step 5, $\bar{p}_r = \frac{\sum_{i=1,m} p_{ir}}{m} > \bar{p}_j = \frac{\sum_{i=1,m} p_{ij}}{m}$ is used for minimizing the number of early

and tardy jobs and $\bar{w}_r=\frac{w_r}{\left\{\frac{\sum_{i=1,m}p_{ir}}{m}\right\}} < \bar{w}_j=\frac{w_j}{\left\{\frac{\sum_{i=1,m}p_{ij}}{m}\right\}}$ for the weighted number of early and tardy jobs.

The time complexity of these algorithms is at most $O(n^2)$. L contains the weights of early and tardy jobs while T contains the JIT jobs.

6. Problem Generation and Computational Results

The preceding heuristics were evaluated on randomly generated problems and compared with an optimal solution, Xpress-MP a commercial software from FICO optimization(2014). The problem formulation given in Section 4 was programmed using the software to give optimal solutions for small instances generated by the procedure given below. The software is designed to give optimal solutions to the Mixed Integer Linear programming. The software is to minimize the (weighted) number of early and tardy jobs on a flow shop machine. Extensive evaluation of their performance were investigated.

6.1 Problem Generation

The heuristics F1 and F2 were tested on problems generated with n = 10, 20, 30, 40, 50, 60, 70 and 80 and the number of machines, m, set at levels (steps) m = 3, 5, 7, 9, 12. Using similar problem instance as Bulfin and M'Hallah (2003), Hariri and Potts (1989) and Gupta and Hariri (1997), two parameters k_1 and k_2 were chosen to provide upper and lower bounds for the due dates. k_1 = {0.2, 0.4, 0.6, 0.8} and k_2 = {0.4, 0.6, 0.8, 1.0}, where $k_1 < k_2$. Let $P = (\sum_{i=1,m}\sum_{j=1,n}p_{ij} + \sum_{j=1,n}(n-1)p_{i*j})$/n be an estimate of the maximum completion time P, obtained by identifying the machine i* with the largest total processing time. The integer earliest due date, a_j, was randomly generated from the uniform distribution [0,Pk_1] and the integer latest due date, d_j, was randomly generated from the uniform distribution [$a_j+\sum_{i=1,m}p_{ij}$, $a_j+\sum_{i=1,m}p_{ij}$ +Pk_2]. The integer weights, w_j, were 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 k_1 and k_2 parameters, ten instances were randomly generated for n = 10, 20 instances were randomly generated for n = 20, e.t.c. For each n and m, 30 replications were generated and the average of the early and tardy weights and time are tabulated. The problem generator and the heuristics were implemented with Java on Eclipse. The two heuristics were compared with results from an optimal solution software, Xpress-MP(TM) on Intel(R) core(TM) i3-3217U CPU with 1.8 GHz, 4 GB RAM. The optimal solution was only able to consider instances of 80 jobs and below. For instances

above 80 jobs, results generated were not consistent and no optimal solution found after 1000 seconds.

6.2 Results

The computational results for both the weighted and unweighted number of early and tardy jobs are found in Tables 1 and 2. Both weights of the early and tardy jobs and the running time in seconds for Algorithms F1, F2 and the optimal solution are presented. For the weighted case in Table 1, wT, indicates the average weighted number of early and tardy jobs on the various machines. As would be expected, the weights (penalties) of the optimal solution are smaller to that of the heuristics. While nT indicates the average number of early and tardy jobs on the various machines.

7. Discussion

Weighted Case

From Table 1, it was observed that the range of the average weighted number of early and tardy jobs for F1 is 35.5334, F2 is 34.4667 and the optimum is 33.75865. The median average weighted number of early and tardy jobs for F1 is 4.86665, F2 is 4.85 and optimum is 3.5837. Similarly, when comparison is made about the minimum average weighted number of early and tardy jobs, F1 is 0.3333, F2 is 0.3333 and optimum is 0.1888 and the maximum average weighted number of early and tardy jobs for F1 is 35.8667, F2 is 35.8 and optimum is 33.94745. In addition, the mean average weighted number of early and tardy jobs for F1 is 7.9675, F2 is 7.911665 and optimum is 7.004014.

The analysis of the running time in seconds for the two heuristics and the optimal are as follow: The range of average time to run the algorithms in seconds for F1 is 0.007933333, F2 is 0.0094333 and optimum is 8.5515. The median average running time in seconds for F1 is 0.004133333, F2 is 0.0039 and optimum is 1.069317. The minimum average running time in seconds for F1 is 0.001, F2 is 0.0005 and optimum is 0.0213. Similarly, the maximum average running time in seconds for F1 is 0.008933333, F2 is 0.0099333 and optimum is 8.5728. In addition, the mean average running time in seconds for F1 is 0.004151667, F2 is 0.0038042 and optimum is 2.051825.

The weighted number of early and tardy jobs generated by the heuristics are not too different from the results produced as optimal solutions. Averagely, heuristic F2 produces better results when compared with heuristic F1. Conversely, F1 takes

lesser running time to complete compared to F2. In all cases considered, the running time for the heuristics are less than a second. It was observed that as the number of jobs increases, the running time of the optimal solutions become exponential.

Unweighted Case

From Table 2, it was observed that the range of the average number of early and tardy jobs for F1 is 6.6333, F2 is 6.6 and the optimum 5.9. The median average number of early and tardy jobs for F1 is 0.96665, F2 is 0.96665 and optimum is 0.6683. Similarly, when comparison is made about the minimum average number of early and tardy jobs, F1 is 0.1, F2 is 0.1 and optimum is 0.01 and the maximum average number of early and tardy jobs for F1 is 6.7333, F2 is 6.7 and optimum is 5.91. In addition, the mean average number of early and tardy jobs for F1 is 1.609995, F2 is 1.601663 and optimum is 1.251663.

The analysis of the running time in seconds for the two heuristics and optimal solutions are as follow: The range of average time to run the algorithms in seconds for F1 is 0.006333, F2 is 0.007833 and optimum is 14.208. The median average running time in seconds for F1 is 0.004167, F2 is 0.0034 and optimum is 1.176417. The minimum average running time in seconds for F1 is 0.001533, F2 is 0.0001033 and optimum is 0.019267. Similarly, the maximum average running time in seconds for F1 is 0.004356, F2 is 0.003926 and optimum is 2.323271. In addition, the mean average running time in seconds for F1 is 0.004356, F2 is 0.003926 and optimum is 2.323271.

The number of early and tardy jobs generated by the heuristics are not too different from the results produced as optimal solutions. As in the weighted case, averagely, heuristic F2 produces better results when compared with heuristic F1. Conversely, F1 takes lesser running time to complete compared to F2. In all cases considered, the running time for the heuristics are less than a second. It was observed that as the number of jobs increases the running time of the optimal solutions become exponential.

Comparison of means of the three algorithms were performed using ANOVA and the results are presented in Table 3, for the weighted number of early and tardy jobs on a flow shop machine and Table 4, the number of early and tardy jobs in a flow shop machine. Statistical analyses in Tables 3 and 4 further show that there is no significant difference in the performance of the heuristics, when F1 and F2 are compared with the results produced as optimal solutions. This indicates that

for large samples where the optimal solution may not be feasible in a realistic time, the heuristics can come handy.

In Figures 1a- 1d, the chart of time performance of the three algorithms are presented for both minimizing the weighted number of early and tardy jobs and minimizing the number of early and tardy jobs for m=3 and m=12. It is observed that the time performances of the optimal solutions are exponential as the number of jobs increases. Furthermore, it can be observed that as the number of machines increases, the time performance of the algorithms reduces, indicating the number of machines is inversely proportional to the time performance of the algorithms in both the weighted and unweighted cases.

Large Instances

The heuristics F1 and F2 were tested on large problem instances with n = 500, 1000, 1500, 2000, 2500 and 3000 and the number of machines, m, set at levels m = 3, 5, 7, 9, 12. Using similar problem instance as indicated earlier, for each n and m, 30 replications were generated and the average of the early and tardy weights and performance times are tabulated. The computational results for both the weighted and unweighted number of early and tardy jobs are found in Table 5. Both weights of the early and tardy jobs and the running time in seconds for F1 and F2 are presented.

Table 1: Comparison of Heuristics with Optimal Solution for Weighted Cases

Number of Machines	Number of Jobs	WEIGHTED					
		wT			Running TIME		
		F1	F2	optima	F1	F2	optima
	10	2.5667	2.3333	2.0988	0.0010	0.0005	0.0214
	20	3.6667	3.7333	3.2878	0.0037	0.0015	0.1177
	30	10.9000	10.6333	9.9533	0.0041	0.0037	0.5557
	40	14.4000	14.0000	13.0855	0.0031	0.0041	0.6683
	50	20.2333	20.2000	19.0510	0.0047	0.0053	2.3876
	60	26.8000	26.6667	25.2832	0.0047	0.0046	4.8581
	70	30.9000	30.4333	28.8153	0.0073	0.0099	6.4134
3	80	35.8667	35.8000	33.9475	0.0089	0.0072	8.5728
	10	1.2667	1.4333	1.1988	0.0021	0.0031	0.0213
	20	3.0000	3.1333	2.6878	0.0020	0.0011	0.1104
	30	6.1000	6.1333	5.4533	0.0016	0.0010	0.5484
	40	6.5667	6.4000	5.4855	0.0026	0.0031	0.6990
	50	7.4000	7.4667	6.3177	0.0041	0.0041	1.3396
	60	11.5000	11.2667	9.8832	0.0042	0.0047	2.2636
	70	15.9333	15.8333	14.2153	0.0042	0.0067	5.9523
5	80	19.6667	19.5667	17.7142	0.0052	0.0042	6.0408
	10	1.1333	1.1333	0.8988	0.0010	0.0026	0.0235
	20	2.7000	2.7000	2.2545	0.0037	0.0026	0.2172
	30	3.6333	3.6333	2.9533	0.0026	0.0036	0.2907
	40	4.0667	4.0667	3.1522	0.0046	0.0032	0.6161
	50	5.7000	5.6667	4.5177	0.0089	0.0036	1.4412
	60	8.3000	8.2667	6.8832	0.0032	0.0021	2.8151
	70	10.1000	10.3333	8.8795	0.0062	0.0037	4.0200
7	80	11.5000	11.0667	9.4487	0.0057	0.0042	5.9013
	10	0.3333	0.3333	0.1888	0.0016	0.0021	0.0255
	20	2.3000	2.3333	1.8878	0.0016	0.0005	0.2494
	30	1.9000	1.8000	1.1200	0.0041	0.0053	0.3345
	40	4.1000	4.1000	3.1855	0.0048	0.0011	0.7551
	50	4.7333	4.7333	3.5843	0.0041	0.0042	1.3641
	60	5.0000	4.9667	3.5832	0.0037	0.0031	2.3116
9	70	6.3000	6.2667	4.6487	0.0036	0.0047	3.3627

	80	7.5667	7.5000	7.2655	0.0067	0.0068	4.1778
	10	0.7000	0.7000	0.5545	0.0026	0.0016	0.0360
	20	1.7000	1.7000	1.5200	0.0037	0.0026	0.2812
	30	1.7333	1.7667	1.6215	0.0031	0.0047	0.3844
	40	2.6000	2.5667	1.1832	0.0042	0.0058	0.8177
	50	3.1333	3.1333	1.5153	0.0052	0.0042	1.3210
	60	3.0667	3.0667	2.8322	0.0042	0.0041	2.2027
	70	3.8333	3.8000	2.6510	0.0073	0.0051	2.9798
12	80	5.8000	5.8000	5.3545	0.0062	0.0057	5.5737

Table 2: Comparison of Heuristics with Optimal Solution for Unweighted Cases

Number of Machines	Number of Jobs	UNWEIGHTED					
		nT			Running Time		
		F1	F2	Optima	F1	F2	Optima
	10	0.5000	0.5000	0.4100	0.0026	0.0010	0.0235
	20	1.1667	1.1667	0.9767	0.0026	0.0026	0.1876
	30	1.9333	2.0000	1.7100	0.0036	0.0021	0.5678
	40	3.0000	2.9333	2.5433	0.0031	0.0058	1.3225
	50	3.9000	3.8000	3.3100	0.0042	0.0057	1.7642
	60	4.9000	4.8667	4.2767	0.0042	0.0052	4.2278
	70	6.3000	6.2333	5.5433	0.0062	0.0068	10.4375
3	80	6.7333	6.7000	5.9100	0.0046	0.0079	14.2273
	10	0.3667	0.3667	0.2767	0.0015	0.0016	0.0193
	20	0.7000	0.7000	0.5100	0.0016	0.0015	0.1552
	30	0.8667	0.8667	0.5767	0.0041	0.0031	0.5402
	40	1.7333	1.7333	1.3433	0.0026	0.0021	0.7219
	50	2.0000	2.0000	1.5100	0.0047	0.0036	1.5209
	60	2.4333	2.3667	1.7767	0.0026	0.0089	3.7274
	70	3.1000	3.1333	2.4433	0.0053	0.0068	6.3451
5	80	3.7333	3.7000	2.9100	0.0062	0.0067	6.8728
	10	0.3000	0.3000	0.2100	0.0042	0.0016	0.0244
	20	0.2333	0.2333	0.0433	0.0026	0.0021	0.2125
	30	0.7333	0.7333	0.4433	0.0042	0.0026	0.2923
	40	0.7667	0.7667	0.3767	0.0031	0.0021	0.6432
	50	1.1333	1.1333	0.6433	0.0058	0.0026	1.5423
	60	1.5000	1.5333	0.9433	0.0031	0.0057	2.1746
7	70	1.9000	1.9000	1.2100	0.0057	0.0042	2.9424

	80	2.5667	2.5000	1.7100	0.0068	0.0052	5.4441
	10	0.1000	0.1000	0.0100	0.0031	0.0026	0.0275
	20	0.4000	0.4000	0.2100	0.0032	0.0026	0.2381
	30	0.6667	0.6667	0.3767	0.0057	0.0030	0.3698
	40	0.9000	0.9000	0.5100	0.0032	0.0046	0.6371
	50	0.9333	0.9333	0.5433	0.0053	0.0057	1.5287
	60	1.0000	1.0000	0.6100	0.0047	0.0047	2.0730
	70	1.1333	1.1333	0.7433	0.0064	0.0047	3.5007
9	80	1.5333	1.5333	1.1433	0.0079	0.0032	4.6664
	10	0.2333	0.2333	0.1433	0.0020	0.0021	0.0354
	20	0.3000	0.3000	0.2100	0.0016	0.0037	0.2870
	30	0.4667	0.4667	0.3767	0.0031	0.0052	0.4011
	40	0.7000	0.7000	0.6100	0.0079	0.0021	0.8225
	50	0.7667	0.7667	0.6267	0.0052	0.0026	1.0304
	60	0.8333	0.8333	0.6933	0.0068	0.0021	1.9381
	70	0.9333	0.9333	0.7933	0.0056	0.0053	3.8215
12	80	1.0000	1.0000	0.8600	0.0073	0.0052	5.6171

Table 3a: Summary: Single Factor for the Weighted case

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
F1	40	318.7	7.9675	69.72401
F2	40	316.4666	7.911665	68.58823
OPTIMA	40	280.1605	7.004014	63.03419

Table 3b: ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	23.403	2	11.7017	0.174352	0.84	3.0738
Within Groups	7852.511	117	67.11548			
Total	7875.914	119				

Table 4a: Summary: Single Factor for the Unweighted case

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
F1	40	64.3998	1.609995	2.530769
F2	40	64.0665	1.601663	2.478511
OPTIMA	40	50.0665	1.251663	1.973572

Table 4b:ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	3.3463	2	1.673144	0.718823	0.49	3.0738
Within Groups	272.3312	117	2.327617			
Total	275.6775	119				

For the weighted case, wT, indicates the average weighted number of early and tardy jobs on the various machines. While nT indicates the average number of early and tardy jobs on the various machines.

Heuristics F2 is slightly better in minimizing the weighted number of early and tardy jobs and the number of early and tardy jobs compared to F1. Conversely, again, the time performance of F1 is better when compared to F2. In some instances, the running time of F2 is relatively higher. In addition, the running time of both heuristics for the weighted and unweighted cases reduces as the number of machines increases. Similarly, for both weighted and unweighted cases, as n increases from 500 to 3,000, so does the time performances of both heuristics increase.

Table 5: Large Instances for Weighted and Unweighted Jobs

		WEIGHTED				UNWEIGHTED			
		wT		CPU RUNNING TIME		nT		CPU RUNNING TIME	
m	n	F1	F2	F1	F2	F1	F2	F1	F2
	500	216.8333	216.7667	0.2167	0.2599	39.2333	39.1333	0.1823	0.2056
	1000	417.3333	416.4667	0.6600	1.1265	76.3333	76.3333	0.5870	0.7240
	1500	636.6667	635.5000	1.6981	2.9101	117.3667	117.3333	1.1778	1.4755
	2000	845.2000	844.6333	3.1189	6.1248	152.3333	152.2667	2.0653	2.6278
	2500	1057.7330	1057.0330	3.4684	9.7579	190.6667	190.6000	3.2633	4.2951
3	3000	1255.1330	1254.0330	4.9925	15.2173	231.6333	231.5667	4.6561	6.1811
	500	121.5333	120.9333	0.0911	0.1162	23.0333	23.0000	0.1125	0.1302
	1000	243.6000	243.6667	0.3005	0.5902	43.4333	43.4000	0.3033	0.3709
	1500	355.5667	355.3667	0.6454	1.6235	65.1000	65.1333	0.6151	0.9066
	2000	458.0667	457.2333	1.1147	3.4441	84.6000	84.6000	1.0839	1.5892
	2500	580.6333	580.6333	1.7515	6.2895	104.9000	104.8667	1.6996	2.5606
5	3000	680.4333	679.8000	2.5336	10.1105	127.2000	127.2000	2.4704	3.9992
	500	85.3667	85.3000	0.0657	0.0842	14.3333	14.3333	0.0662	0.0766
	1000	156.9333	156.5333	0.1974	0.3912	28.6000	28.6000	0.1959	0.2625
	1500	223.9333	223.8667	0.4167	1.0668	40.8667	40.8667	0.3985	0.5787
	2000	298.4333	297.7333	0.6960	2.4065	54.7667	54.7667	0.7078	1.0710
	2500	372.0333	371.8667	1.0791	4.3111	67.7333	67.7333	1.0849	1.7997
7	3000	441.1000	440.9333	1.6478	7.0831	80.6667	80.6333	1.5959	2.7846
	500	57.3000	57.2333	0.0467	0.0616	10.6667	10.6667	0.0506	0.0587
	1000	112.3667	112.3000	0.1507	0.2943	20.1000	20.1000	0.1453	0.1974
	1500	175.5000	175.2667	0.3276	0.8872	29.6667	29.6667	0.2974	0.4686
	2000	216.8667	216.6000	0.4972	1.7731	38.3667	38.3333	0.4918	0.8225
	2500	255.6333	255.2667	0.8080	3.0632	49.0000	48.9333	0.7515	1.3636
9	3000	310.4000	310.2667	1.1275	5.1263	58.8000	58.8000	1.1365	2.1209
	500	38.8667	38.8000	0.0512	0.0557	7.7000	7.7000	0.0490	0.0494
	1000	73.6667	73.4000	0.1229	0.2115	13.4333	13.4333	0.1219	0.1521
	1500	108.4333	108.4333	0.2502	0.5693	20.2000	20.2000	0.2203	0.3559
	2000	143.8333	143.8000	0.3713	1.1589	26.3000	26.3000	0.3662	0.6651
	2500	179.0667	178.7667	0.5907	2.1328	33.5000	33.5000	0.5927	1.0511
12	3000	209.9667	209.5667	0.8015	3.3074	37.2000	37.2000	0.7996	1.4536

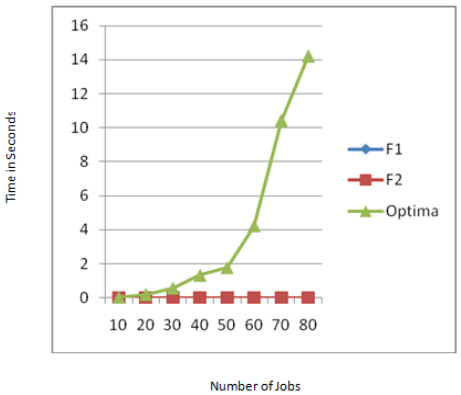

Figure 1a: Chart of Time Performance of the Heuristics and Optima for M=3 in Unweighted Case

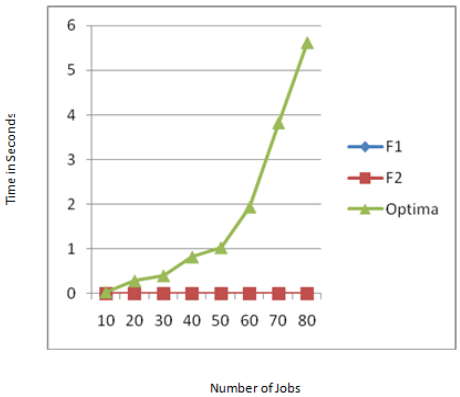

Figure 1b: Chart of Time Performance of the Heuristics and Optima for M=12 in Unweighted Case

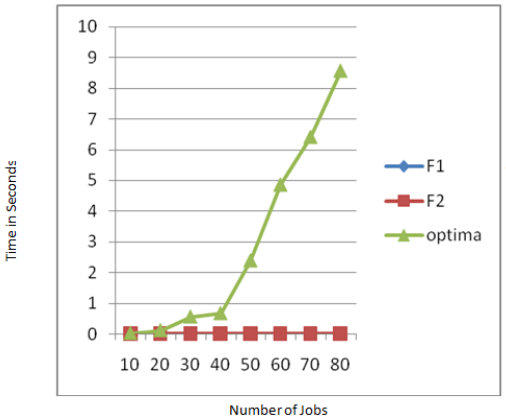

Figure 1c: Chart of Time Performance of the the Heuristics and Optima for M=12 in Weighted Case

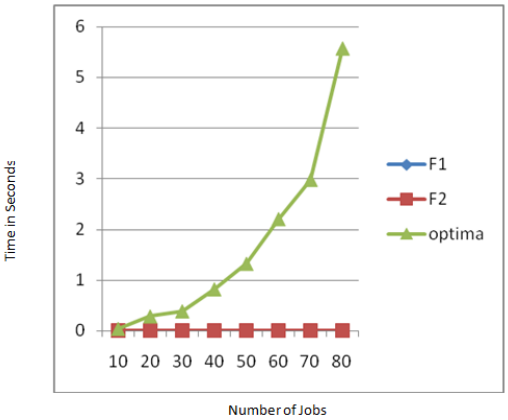

Figure 1d: Chart of Time Performance of Heuristics and Optima for M=3 in Weighted Case

8. Conclusion

In this paper, 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. It is known that this problem is NP Complete for when the due date is at a point in time, indicating no efficient optimal solution in a reliable time for even small instance problems. In this research, two greedy heuristic solutions are compared with an optimal solution. Computational results and analysis for various scales of instances show that the greedy heuristic algorithms performed creditably well when compared with an optimal solution using small problem instances. The quality and efficiency of the heuristics coupled with solutions to large instance problems are highlighted. Further research should seek to improve on these results by using Meta-heuristics methods. In addition, approximation and pseudo-polynomial algorithms could be developed.

References

Adamu M.O and Adewumi A.O., 2014. Single Machine Review to Minimize Weighted Number of Tardy Jobs. *Journal of Industrial and Management Optimization.*10(1), pp 219–241.

Adamu M.O and Adewumi A.O. (2016). Minimizing the Weighted Number of Tardy Jobs on Multiple Machines: A Review. *Journal of Industrial and Management Optimization*. 12(4), pp 1465- 1493.

Adamu, M.O., Budlender, N. and Idowu, G.A. (2014). A Note on Just-In-Time Scheduling on Flow Shop Machines. *Nigerian Mathematical Society*, **33**: 321-331.

Baki, M.F. and Vickson, R.G. (2004). One Operator, Two Machine Open Shop and Flow Shop Problems with Setup Times for Machines and Weighted Number of Tardy Jobs Objective. *Optimization Methods and Software*, **19** (2): 165-178.

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

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

Croce, F.D., Gupta, J.N.D. and Tadei, R. (2000). Minimizing Tardy Jobs in a Flowshop with Common Due Date. *European Journal of Operational Research*, **120**: 375-381.

Desprez, C., Chu, C. and Chu, F. (2006). A Genetic Algorithm for Minimizing the Weighted Number of Tardy Jobs. *Proceedings of IEEE*: 1271-1276.

FICO optimization(2014). Xpress-MP Version 7.7

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

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

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

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

Ho, J.C. and Gupta, J.N.D. (1995). Flowshop Scheduling with Dominant machine. *Computers and Operations Research*, **22**: 237-246.

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

Lawler, E.L. and Moore, J.M. (1969). A Functional Equation and its Applications to Resource Allocation and Sequencing Problems. *Management Science*, **16**: 77-84.

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

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

Lin, B.M.T. (2001). Scheduling in the Two-Machine Flow Shop with Due Date constraints. *Journal of Production Economics*, **70**: 117- 123.

Shabtay, D. and Bensoussan, Y. (2012). Maximizing the Weighted Number of Just-In-Time Jobs in Several Two Machine Scheduling Systems. *Journal of Scheduling*, **15**(1): 39–47.

Ucar, H. and Tasgetiren, M.F. (2006). A Particle Swarm Optimization Algorithm for Permutation Flow Shop Sequencing Problem with the Number of Tardy Jobs Criterion. *Proceedings of 5th International Symposium of Intelligent Manufacturing Systems*, Sakarya, Turkey: 1110-1120.

Xiang, S., Tang, G. and Cheng, T.C.E. (2000). Solvable Cases of Permutation Flowshop Scheduling with Dominating Machines. *International Journal of Production Economics*, **66**: 53-57.

Yeung, W.K., Oğuz, C. and Cheng, T.C.E. (2009). Two-Machine Flow Shop Scheduling with Common Due Window to Minimize Weighted Number of Early and Tardy Jobs. *Naval Research Logistics*, **5**: 593-599.

Acknowledgement

The authors are grateful to the referee(s) for their useful comments that improved the quality of this paper.