International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2015 , pp. 1-15

Minimizing the Weighted Number of Early and Tardy Jobs on Uniform Machines

Muminu O. Adamu[1*], Gbolahan Idowu [2]

1. Department of Mathematics, University of Lagos, Akoka, Yaba, Lagos.
2. Department of Computer Science, Lagos State University, Ojo, Lagos.
*Corresponding author: madamu@unilag.edu.ng

Abstract

Scheduling to maximize the (weighted) number of Just-In-Time (JIT) jobs or minimize the (weighted) number of early and tardy jobs on uniform parallel machines are considered in this paper. The mathematical model formulation for the general case is presented. It is known that these problems are NP Complete on a single machine; this suggests that no efficient optimal solution seeking algorithms can obtain solution at polynomial time as the problem size increases. Two greedy based heuristic algorithms are proposed for solving the problem with a numerical example to illustrate it use. Extensive computational experiments performed with the heuristic on large scale problem sizes showed promising results.

1 Introduction

Scheduling is concerned with the problem of assigning a set of jobs to a set of machines subject to a set of constraints. Scheduling constraints include due date (e.g. a date when a job must be completed), priority on jobs (e.g. finish a job as soon as possible while meeting the other due dates) and machine capacities (e.g. the speed of each machine). Scheduling problems entail solving for optimal schedule with various objectives, different machine environments and characteristics of the jobs. [6] classified scheduling problem with a three field representation $\alpha|\beta|\gamma$ (α – machine environment, β – Job characteristic and γ – Objective of interest).

Uniform parallel machines can be characterized as machines with different speed factor, and each job has a single operation. A manufacturing unit can invest in machines that have same capacity considering operational cost. The reality that machines advance in technology and are bought at different dates is also a reason for getting "uniform" machines. The problem of scheduling jobs on uniform machines to maximize the (weighted) number of tardy jobs have been done by [8], [9], [2], [13]. They studied uniform parallel machines with single due dates for each job.

IJMAO

International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2015 , pp. 1-15

This paper considers independent jobs to be scheduled on machines which are available for processing simultaneously, each with an interval due dates rather than single due dates, called due window of each job. The earliest due date $a_j \geq 0$ (i.e. the instant at which a job can be completed and delivered), and the latest due date $d_j \geq 0$ (an instant by which processing or delivery of a job must be completed). No penalty is incurred when a job is completed within its due window, but earliness (tardiness) penalty is incurred if a job is completed before its earliest due date or after its latest due date. Pre-emption of jobs is not allowed (e.g when a job begins on a machine it must be allowed to complete its operation). Adopting the problem classification of [6], the dual of the problems become $Qm||\sum w_j(U_j+V_j)$ and $Qm||\sum(U_j+V_j)$ (i.e., minimizing the (weighted) number of early and tardy jobs on m uniform parallel machines). However, the initial problems are maximizing the (weighted) number of JIT jobs on uniform machines.

JIT has numerous application in chemical or high technology industries where parts must be ready at specific times in order to meet certain required conditions (such as arrival of other parts, specific temperature, pressure, etc.). Production of perishable items (e.g. food, drugs, and photographic films) under deterministic demands having similar cost structure. Other applications can be found in production units with no or limited capacity for storage where the due windows are determined by the pick-up times, and pick-ups are made by customers. If the due window (pick-up) is missed, a special delivery service needs to be bought by the producer, the cost of which is dependent on the earliness/tardiness penalty. Rental agencies (hotels, car rentals) plan reservation schedule to meet exactly the request times of customers. For instance, if a customer requires a room (or car) within specific dates, but the dates could not be met, the customer would look for an alternative accommodation/agency, with loss of income for the agency. Alternatively, the agency could offer a deal whereby the customer would be scheduled on a different date with compensation. The objective is to maximize the number of customers scheduled as requested.

The remaining parts of the paper are as follows: Section 2 considers the review of relevant literature. The problem formulation is outlined in section 3. The heuristic algorithms for our problems are presented in section 4. In section 5, the problem generation and computational results are enumerated and finally, in section 6, the concluding remarks are given.

2 Literature Review

The parallel machine scheduling problem can be classified into three categories: identical parallel machines scheduling problem, uniform (proportional) parallel machines scheduling problem and the unrelated parallel machines scheduling problem. A complete review of literature can be found in [1]. [10] have shown that our problem is NP Complete in the strong sense even on a single machine.

One of the earliest to minimize the number of tardy jobs on uniform machines

IJMAO
International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2015 , pp. 1-15

was [8] who considered $Qm|pmtn|\sum U_j$ and gave a polynomial complexity time $O(n^4)$ for $m = 2$ and $O(n^{3(m-1)})$ for $m \geq 3$. [9] provided an improved $O(n^3)$ algorithm for the special case of $m = 2$, making the time bound $O(n^{3(m-1)})$ uniform for all m. For the $Qm|pmtn|\sum w_jU_j$ they proposed a $O(Wn^2)$ time, where W is the sum of the job weights, i.e. minimizing the weighted number of late jobs. They also presented a fully polynomial approximation scheme for the weighted case.

[2] considered the problem $Q|pmtn, pj = p|\sum U_j$ and proposed a solution in time $O(n\log^2 n + mn\log n)$.

[4] studied the problem $Q||\sum w_jU_j$. Using Dantzig-Wolfe decomposition, they reformulated it as a set of partitioning problem. They constructed a branch and bound algorithm through column generation. The average time to solve 10 machine 100 job instances was 1.59 hours for the $Q||\sum w_jU_j$. [11] used bounds from a surrogate relaxation resulting in a multiple knapsack. Extensive computational experiments compared with [4] showed better timing.

[13] studied the parallel machines where the speeds of the machines depend on the allocation of a secondary resource. They gave two versions of the problem of minimizing the number of tardy jobs. The first version assumes that the jobs are pre-assigned to the machines, while the second one takes into consideration the task of assigning jobs to the machines. They proposed an integer programming formulation to solve the first case and a set of heuristics for the second.

[5] considered the problem of scheduling n identical jobs on m uniform parallel machines to optimize scheduling criterion of minimizing the weighted number of tardy jobs. They gave an $O(n\log n)$ algorithm for solving the problem.

3 Problem Formulation

For any given schedule S, let p_j, t_{ij} and $C_j(S) = t_{ij}+p_j$ represent the processing time, actual start time on a given machine and completion time of job j on machine i, respectively. Job j is said to be early if $C_{ij}(S) < a_j$, tardy if $C_{ij}(S) > d_j$ and on-time if $a_j \leq C_j(S) \leq d_j$. In addition, let x_{ij} for j be defined as follows:

$$x_{ij} = \begin{cases} 1 & \text{if } a_j \leq C_S \leq d_j \text{ (On-time)} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

Furthermore, let $w_j \geq 0$ be the weights for the early/tardy jobs. The mathematical model of this problem is given as follows:

$$P = \max \sum_{i=1}^{m}\sum_{j=1}^{n} w_j x_{ij} \tag{3.2}$$

IJMAO
International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2015 , pp. 1-15

Subject to:

$$a_j \leq \max_{k=1}^{j}\{\{C_{ik-1}(S), a_j - p_j\} + p_j\}x_{ij} \quad i = 1, \ldots, m; \quad j = 1, \ldots, n. \tag{3.3}$$

$$\max_{k=1}^{j}\{\{C_{ik-1}(S), a_j - p_j\} + p_j\}x_{ij} \leq d_j \quad i = 1, \ldots, m; \quad j = 1, \ldots, n. \tag{3.4}$$

$$\sum_{i=1}^{m} x_{ij} \quad j = 1, \ldots, n. \tag{3.5}$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \ldots, m; j = 1, \ldots, n \tag{3.6}$$

Equation (3.2) is the objective function of the general case, that is, maximizing the weighted number of JIT jobs on the uniform machines. Constraint (3.3) insures job j is not finished before its earliest start time a_j and $C_{i,j-1}$ is the time to complete the $(j-1)th$ job on machine i. Similarly, C_{ij} is the time to complete the jth job on machine i. Constraint (3.4) is the completion time of job j if it is no greater than it's latest due date d_j. Constraint (3.5) insures that a job is assigned to at most one machine, and constraint (3.6) forces a job to be either on-time or early/tardy; 1 if on-time and zero otherwise.

4 Heuristic

In this section, the heuristic algorithms proposed for solving the problems of minimizing the (weighted) number of early and tardy jobs on uniform machines are presented. The differences between the two heuristics proposed are indicated within the heuristic given below. In step 3, we break tie by smallest p_{ij} for heuristic minimizing the number of early and tardy jobs and highest $\frac{w_j}{p_{ij}}$ for heuristic minimizing the weighted number of early and tardy jobs. Similarly, in steps 3 and 4, $p_r > p_j$ is used for heuristic minimizing the number of early and tardy jobs while $\frac{w_r}{p_r} < \frac{w_j}{p_j}$ is used for heuristic minimizing the weighted number of early and tardy jobs on the uniform machines.

4.1 Heuristic for Scheduling on Uniform Machines

1. Re-index the jobs on the machines such that $a_1 \leq a_2 \leq \ldots \leq a_n$; $T_i := \emptyset$; $L_i := \emptyset$; $Q := \{J_1, J_2, \ldots, J_n\}$; $t_{i0} := 0$; $i := 1, 2, \ldots, m$; $|T_i| := 0$;$j := 0$; $L := \sum_{i=1}^{m} L_i$
2. Arrange machines in order of decreasing speed
3. Assign the jobs to the machines in order of decreasing speed and break tie by smallest p_{ij} or highest $\frac{w_j}{p_{ij}}$

   for $i := 1$ to m **do**
   for $j := 1$ to n **do**
   if $\max\{t_{i,j-1}, a_j - p_{ij}\} + p_{ij} \leq d_j$ **then**
   $T_i := T_i \cup \{J_j\}$; $Q := Q \setminus \{J_j\}$; $j := j + 1$; $|T_i| := |T_i| + 1$
   else

IJMAO
International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2015 , pp. 1-15

end if
end for
end for

4. Reassign remaining jobs in Q on the machines if they could be scheduled ontime without making any scheduled job late. ($Q := J_1, J_2, \ldots, Jq$)
for $i := 1$ to m **do**
for $p := 1$ to q **do**
Find jobs J_r in T_i with $p_r > p_j$ or $\frac{w_r}{p_r} < \frac{w_j}{p_j}$
for $l := 1$ to $|J_r|$ **do**
Remove Job J_k from T_i
Reassign Job J_q from Q into T_i
if $\max\{t_{ij-1}, a_j - p_{ij}\} + p_{ij} \leq d_j$ **then**
$T_i := T_i \cup \{J_q\}$; $Q_i := Q_i \setminus \{J_q\}$
$L_i := L_i \cup \{J_k\}$;
else
end if
end for
end for
end for
Move all Q into L

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

4.2 Numerical Example

A numerical example of ten jobs to minimize the weighted number of early and tardy jobs on 3 uniform machines is presented in Table 1.

Table 1: Data for the Numerical Example

Job	a_j	d_j	Processing time (M_1)	Processing time (M_2)	Processing time (M_3)	Weight w_j	$\frac{W_j}{p_{ij}}$
1	3	27	8	12	24	3	
2	4	18	4	6	12	2	
3	6	36	10	15	30	1	
4	1	31	10	15	30	5	
5	5	24	6	9	18	1	
6	8	20	4	6	12	1	M_3=0.0833
7	11	35	8	12	24	2	M_1=0.25;M_2=0.1667
8	4	10	2	3	6	2	
9	1	19	6	9	18	1	M_1=0.1667;M_2=0.111
10	7	44	12	18	36	1	

Let L be the set of late jobs and Q the set of scheduled jobs. t_{ij-1} is the time of the previous job on machine i.

IJMAO
International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2015 , pp. 1-15

Illustration for the Uniform Machines

Step 1: Assign the jobs according to their earliest due dates.
$[J_4, J_9, J_1, J_2, J_8, J_5, J_3, J_{10}, J_6, J_7]$

Step 2: Arrange the jobs on these machines in order of their speed.
Machine 1:
Machine 2:
Machine 3:

Step 3: Assign the jobs to the various machines and when there is a tie, break tie by highest $\frac{W_j}{p_{ij}}$ (HPWJ)

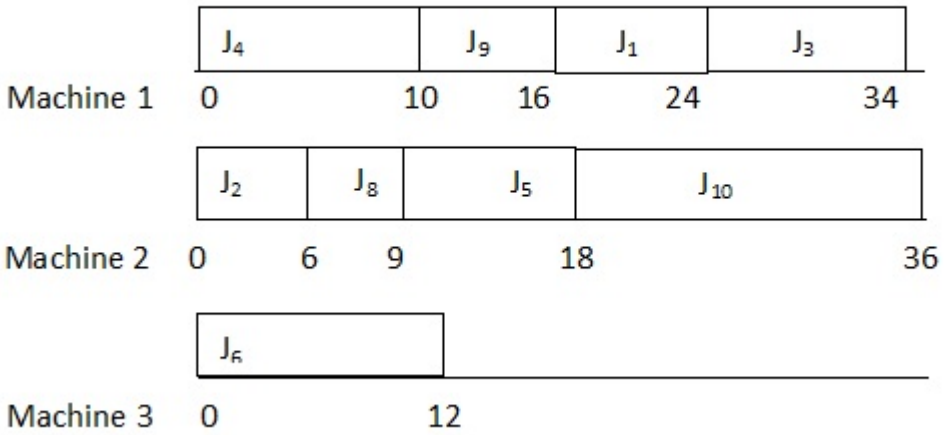

Figure 1: Gantt Chart of the numerical example at step 3

at step 3, Job J_7 is Tardy (Late)

Step 4: Reassign jobs in Q (Unscheduled jobs) on the Machines if they could be scheduled on-time without making any scheduled job late.
The job J_7 tardy (Late) is tried on machines 1 to see if possible to be on-time. It can be scheduled on machine 1 without affecting already scheduled jobs, replacing Job J_9
(Since $\frac{w_7}{p_{17}} > \frac{w_9}{p_{19}}$). Job J_9 is again tried on machines 2 & 3.
Incidentally, it remains tardy.
Job J_9 is Tardy(Late)

Step 5: Stop

International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2015 , pp. 1-15

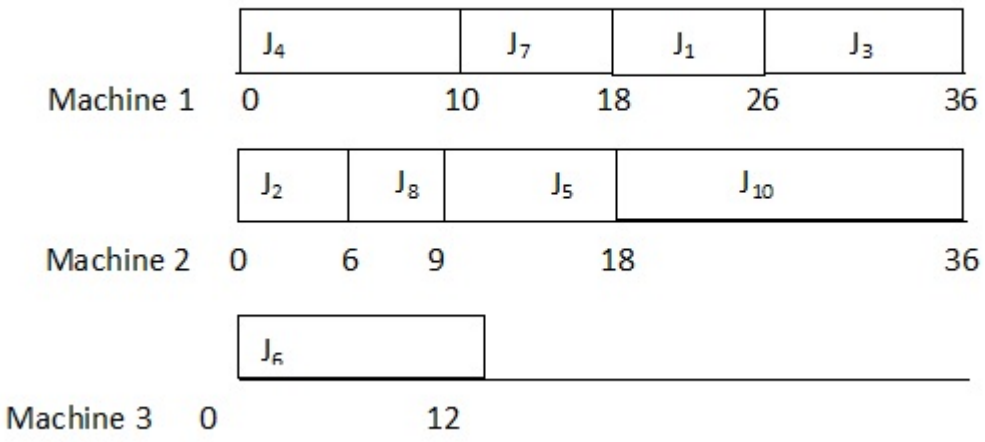

Figure 2: Gantt Chart of the numerical example at step 4

5 Problem Generation and Computational Results

In this section, the above heuristics were evaluated on randomly generated problems and extensive computational analysis of their performance investigated.

5.1 Problem Generation

Experiments were designed to test the relative effectiveness of the proposed algorithms. The heuristics would be tested on problems with 100, 200, 300, 400 and 500 jobs that would be generated as in [7], [3], [12], and [14]. The number of machines is set at six levels: 2, 4, 6, 8, 10 and 12. For each job j, an integer processing time p_j is randomly generated in the interval in [1,99]. Two parameters k_1 and k_2 are used, and taken in the set $\{1, 2, 3, 4\}$. For the data to depend on the number of jobs n, the integer earliest start time a_j is randomly generated in the interval $[0, \frac{nk_1}{m}]$ and the integer latest due date d_j is randomly generated in the interval $[a_j + p_j, a_j + \frac{2n\bar{p}k_2}{m}]$ where $\bar{p} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}}{m}$. For each combination of n, k_1 and k_2, 10 instances are generated, i.e., for each value of n, 160 instances are generated with a weight randomly chosen in [1, 10] and 8,000 problems (50 replications) where m is the number of machines. k_1 and k_2 indicates the different levels of Traffic Congestion Ratio (TCR). The larger the value of k_1 and k_2 the more congested the queue will be, and the higher the number of tardy jobs will result. The heuristics are implemented with Java on a Intel(R) Core(TM) i3-3217U CPU with 1.8GHz, 4 GB RAM.

IJMAO
International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2015 , pp. 1-15

5.2 Computational Result

The computational results for both minimizing the weighted number of early and tardy jobs and the number of early and tardy jobs are presented in Table 2. The weighted number of jobs, number of jobs and their respective running times in seconds are shown. For the weighted case in Table 2, avegWT, indicates the average weighted number of early and tardy jobs on various uniform machines; while avegnT indicates the average number of early and tardy jobs on the various machines.

From Table 2, it is observed that the range of the average weighted number of early and tardy jobs is 117.48 where the minimum is 0 and maximum is 117.48 for the heuristic used. The median average weighted number of early and tardy jobs is 3.62. The mean average weighted number of early and tardy jobs is 13.61.

The observed running times in seconds of the heuristic for the problem of minimizing the weighted number of early and tardy jobs are as follow: Range (6.01), Minimum (3.56), Maximum (9.57), Mean (6.56) and Median (6.97)

Similarly, for the problem of minimizing the number of early and tardy jobs on the uniform machines, the range is 22.67 where the minimum is 0 and maximum is 22.67. The median average number of early and tardy jobs is 1.55. The mean average number of early and tardy jobs is 3.24. The observed running times in seconds of the heuristic for the problem of minimizing the number of early and tardy jobs are as follow: Range (14.33), minimum (9.74) maximum (17.14), mean (9.74) and median (9.53).

It is observed for both problems considered that as n increases the (weighted) number of early and tardy jobs decreases. The reason is due to the number of jobs in the experiment being constant. Similarly, as the number of machines increases, the running times of the heuristics also increase. Indicating that more machines increase the running times of the heuristics even when the number of jobs is kept constant. In Figures 3 and 5, the performance of the heuristics is shown for when $m = 2$ and $m = 10$ for both problems under consideration. The charts show that the (weighted) number of jobs reduces with increase in the number of jobs scheduled. Figures 4 and 6 reveal the running time in seconds of the heuristics for when $m = 2$ and $m = 10$ for both problems. From the charts, it is evident that as the number of jobs increases so does the running time of the heuristics.

Similarly, Figures 7 and 9 show the performance of the heuristics for when $n = 100$ and $n = 500$ for both problems under review. They indicate that the (weighted) number of jobs decreases with the increase in the number of machines. In conclusion, Figures 8 and 10 show the time performance in seconds of the heuristics for when $n = 100$ and $n = 500$. They also indicate that the running time of the heuristics increase with the increase in the number of machines. However, unlike in Figures 4 and 6, the running times of the heuristics for both problems considered are very close, indicating that the weights of the jobs have little effect on the timing of the heuristics when compared with the equal weighted jobs problem.

IJMAO
International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2015 , pp. 1-15

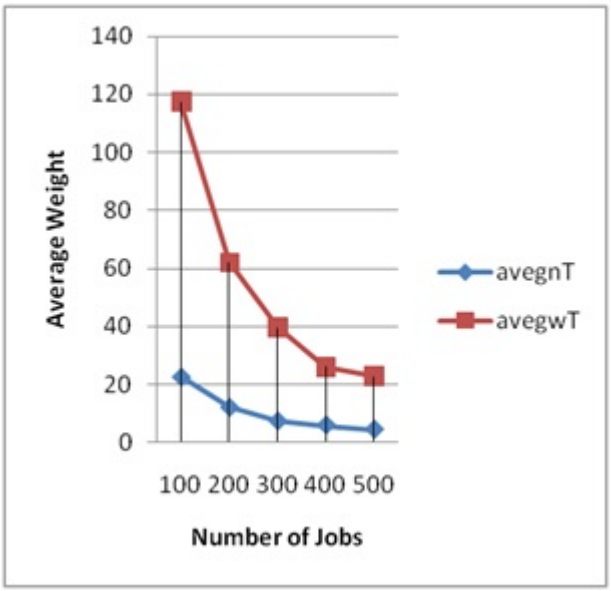

Figure 3: Chart of Heuristic Performance when m=2

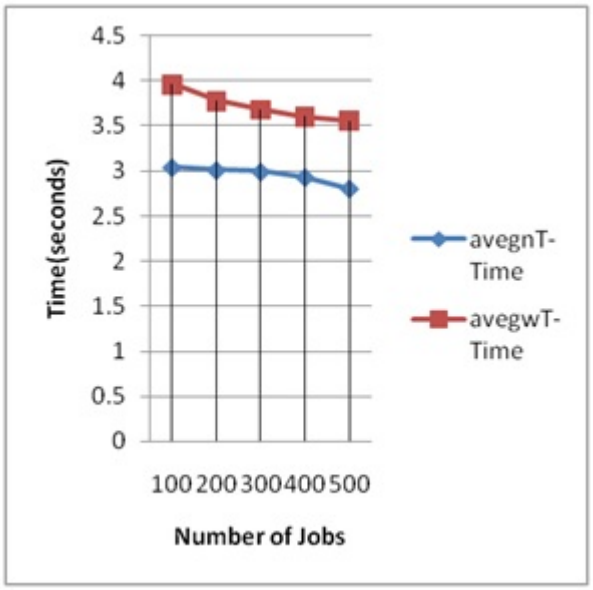

Figure 4: Chart of Time Performance of the Heuristic when m=2

IJMAO
International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2015 , pp. 1-15

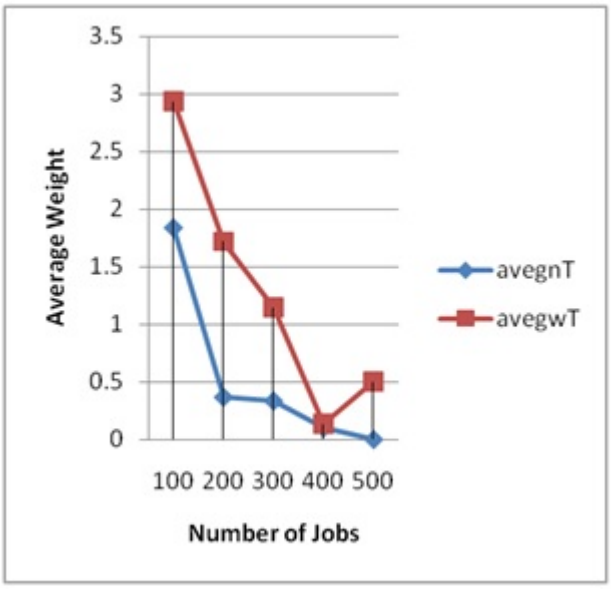

Figure 5: Chart of Heuristic Performance when m=10

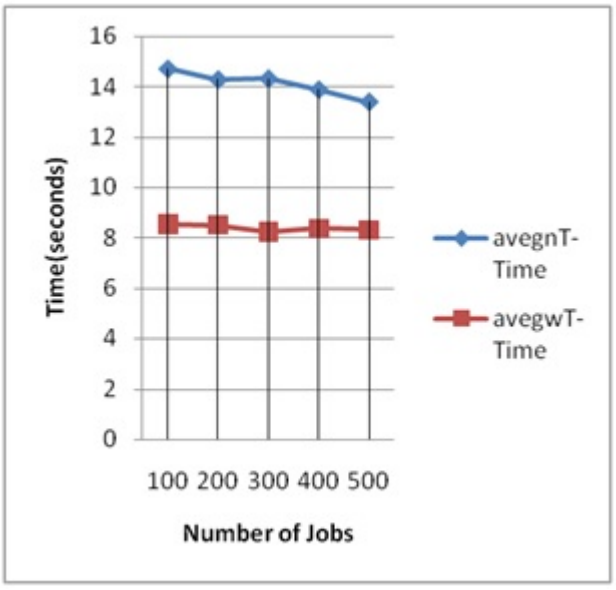

Figure 6: Chart of Time Performance of the Heuristic when m=10

International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2015 , pp. 1-15

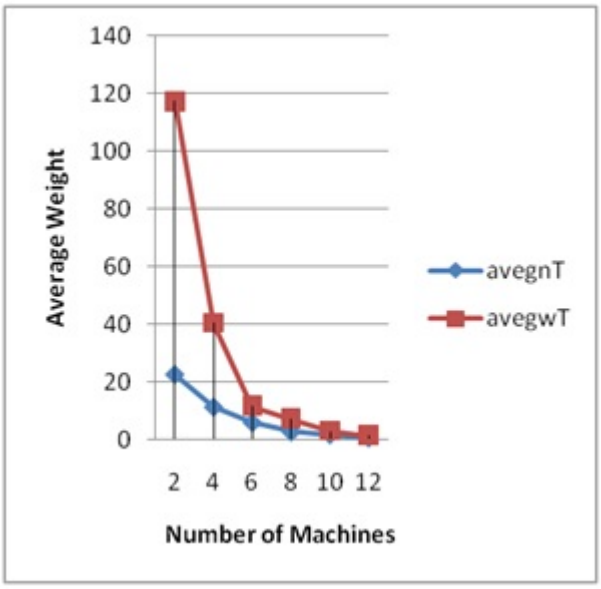

Figure 7: Chart of Heuristic Performance when n=100

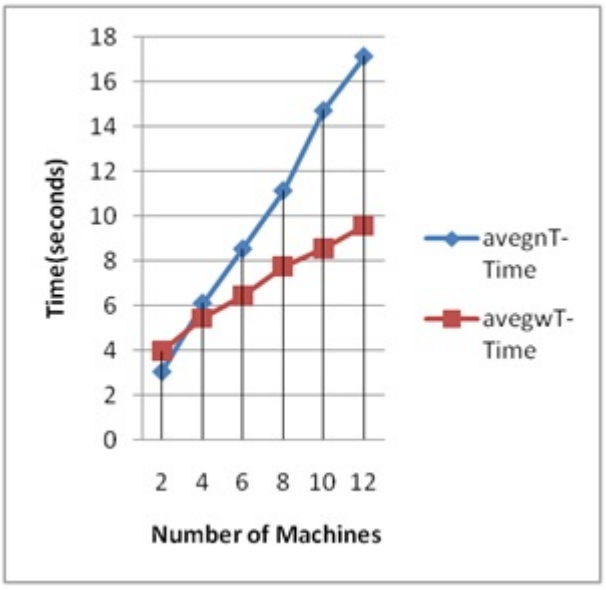

Figure 8: Chart of Time Performance of the Heuristic when n=100

IJMAO
International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2015 , pp. 1-15

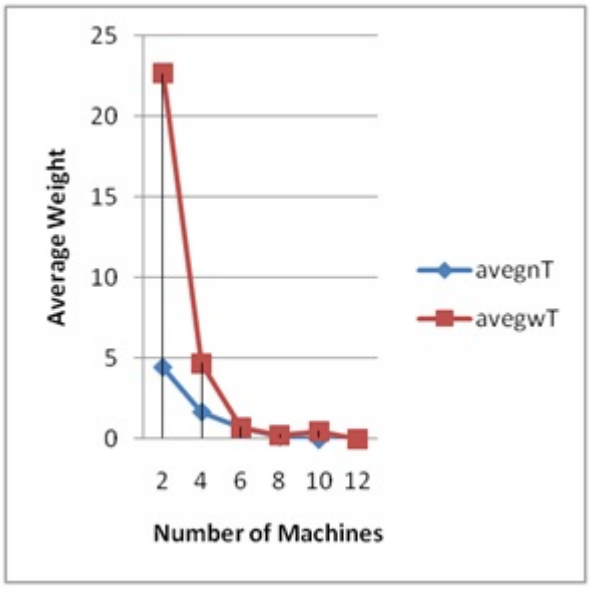

Figure 9: Chart of Heuristic Performance when n=500

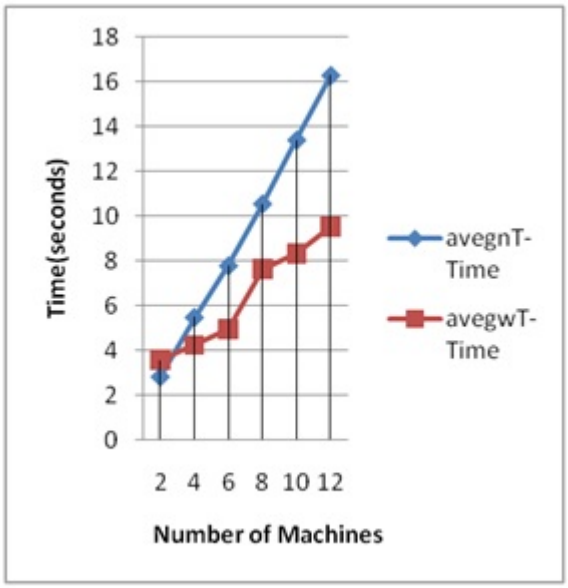

Figure 10: Chart of Time Performance of the Heuristic when n=500

IJMAO
International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2015 , pp. 1-15

Table 2: Results for (Weighted) number of early and Tardy jobs

m	n	Unweighted Case		Weighted Case	
		avegnT	avegTime	avegwT	avegTime
2	100	22.6667	3.040367	117.48	3.96466
2	200	12.1	3.008633	61.9	3.77856
2	300	7.3333	2.991367	39.48	3.6861
2	400	5.9667	2.9292	25.98	3.60426
2	500	4.4667	2.801333	22.68	3.55794
4	100	11.4	6.104033	40.5	5.428375
4	200	4.9667	5.599167	19.0588	4.650882
4	300	3.2	5.581633	12.7188	4.5464
4	400	2.3333	5.545567	8.3947	4.295531
4	500	1.7	5.452367	4.68	4.225026
6	100	6	8.528467	11.6667	6.442667
6	200	2.6667	8.186733	9.5882	5.683176
6	300	1.4667	8.059433	3.3529	5.457
6	400	1	7.952667	3.88	5.31076
6	500	0.7	7.759367	0.75	4.930571
8	100	2.7667	11.13517	7.25	7.72075
8	200	1.6333	11.17307	5.7391	7.620522
8	300	0.5333	11.03833	3	7.607536
8	400	0.5333	10.7309	0.8667	7.499933
8	500	0.1667	10.52823	0.2759	7.60369
10	100	1.8333	14.7139	2.9333	8.533067
10	200	0.3667	14.28397	1.7143	8.495095
10	300	0.3333	14.3193	1.1429	8.216
10	400	0.1	13.87313	0.1333	8.354733
10	500	0	13.38207	0.5	8.299167
12	100	0.7407	17.13615	1.3158	9.570316
12	200	0.1852	17.10522	0.7931	9.376552
12	300	0.0333	16.57147	0.3793	9.435241
12	400	0.0333	16.37203	0	9.291333
12	500	0	16.27043	0	9.511833

It takes more time to run a couple (m,n) of the weighted case when compared with the unweighted case. For example, when $m = 2$ and $n = 100$, the average time for the weighted case on the Uniform machines is 6.56 seconds compared to 9.74 for the unweighted case.

6 Conclusion

In this paper, the problems of scheduling to maximize the (weighted) number of JIT jobs on uniform machines were considered. The dual of these problems are minimizing the (weighted) number of early and tardy jobs on uniform machines. A mathematical model formulation for the general case, that is, the weighted case was provided. Two greedy heuristic algorithms were proposed for solving these problems. A numerical example for illustrating the heuristic for the general case was presented. The problem is NP-complete therefore suggest that computational time will increase exponentially with problem size. Computational experiments were demonstrated with results showing the effectiveness of the heuristics for large problem size. The results and analyses revealed that the heuristics are promising. Further research should seek to improve on these results by using evolutionary algorithms, find exact solutions for small samples where possible, evolve approximation and pseudo-polynomial algorithms.

References

[1] Adamu, M.O. & Adewumi, A. A Survey of Single machine Scheduling to Minimize Weighted Number of Tardy Jobs. *Journal of Industrial and Management Optimization* **10**, 3, 219–241 (2014).

[2] Baptiste, P., Brucker, P., Knust, S. & Timkovsky, V.G. Ten Notes on Equal Processing Time Scheduling. *Quarterly Operation Research* **2**, 111–127 (2004).

[3] Baptiste, P., Peridy, E. & Pinson E. A branch and bound to minimize the number of late jobs on a single machine with release time constraints. *European Journal Operational Research* **144**, 1–11 (2003).

[4] Chen, Z., & Powel, W.B., Solving Parallel Machine Scheduling Problems by Column Generation. *INFORMS Journal on Computing* **11** (1), 78–94 (1999).

[5] Dessouky, M.J., Lageweg, B.J.,Lenstra, J.K., & Vande Velde, S.L., Scheduling Identical Jobs on Uniform Parallel Machines. *Statistica Neerlandica* **44** (3), 115–123 (1990).

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

[7] Ho, J.C. & Chang, Y.L., Minimizing the number of tardy jobs for m parallel machines. *European Journal of Operational Research* **84**, 343–355 (1995).

[8] Lawler, E.L., Effecient Implementation of Dynamic Programming Algorithms for Sequencing Problems. *Report BW 106/79, Math. Centre, Amsterdam* (1979).

[9] Lawler, E.L. & Martel, C.U., Preemptive Scheduling of Two Uniform Machines to Minimize the Number of Late Jobs. *Operations Research* **37** (2), 314–318 (1989).

[10] Li, C.L., Cheng, T.C.E. & Chen, Z.L., Single machine scheduling to minimize the weighted number of early and tardy agreeable jobs. *Computers and Operations Research* **22** (2), 205–219 (1995).

[11] M'Hallah, R. & Bulfin, R.L., Minimizing the weighted number tardy jobs on parallel processors. *European Journal of Operational Research* **160** (2), 471–484 (2005).

[12] Potts, C.N. & Wassenhove, L.N., Algorithms for scheduling a single machine to minimize the weighted number of late jobs. *Management Science* **34** (7), 843–858 (1988).

[13] Ruiz-Torres, A.J., Lopez, F.J. & Ho, J.C., Scheduling Uniform Parallel Machines Subjected to a Secondary Resource to Minimize the Number of Tardy Jobs. *European Journal of Operational Research*, **179**, 302–315 (2007).

[14] Sevaux, M. & Dauzere-Peres, S., Genetic algorithms to minimize the weighted number of late jobs on a single machine. *European Journal of Operational Research* **151**, 296–306 (2003).