HEURISTICS FOR MINIMIZING THE NUMBER OF TARDY JOBS ON A SINGLE MACHINE WITH RELEASE TIME

E. O. Oyetunji[1] and **A. E. Oluleye**[2]

[1,2]Department of Industrial and Production Engineering
University of Ibadan, Nigeria
[1]eoyetunji@yahoo.com, [2]ayodeji.oluleye@mail.ui.edu.ng

ABSTRACT

This paper considers the problem of minimizing the number of tardy jobs with release time on a single machine. Given that the problem has been classified as strongly NP-Hard, three heuristics (EOO, HR2, and HR3) are proposed for this problem. They are compared with a heuristic by Dauzere-Perez (selected from the literature). Randomly-generated problems ranging from 3 to 500 jobs are solved. Experimental results show that one of the proposed heuristics (EOO) outperforms other heuristics, both in terms of quality of solution (effectiveness) and speed of execution (efficiency).

OPSOMMING

Die navorsing behandel die ministering van voltooiingstyd van die aantal draaltake by ‘n enkele werktuig. As aanvaar word dat die problem geklassifiseer word as hoofsaaklike NP-hard, word voorgestel dat die vraagstuk bestudeer word deur gebruik te maak van drie heuristiese metodes (EOO, HR2, HR3). Die metodes word vergelyk ten opsigte van vertoning met die Dauzere-Perez-metode. Toevalsgegenereerde probleme wat strek vanaf 3 tot 500 draaltake word behandel. Die eksperimentele werk lewer bewys dat die EOO-metode ander metodes die loef afsteek ten opsigte van oplossingsgoedheid en –snelheid.

1. INTRODUCTION

Scheduling is concerned with the problem of allocating resources (machines) over time to perform a number of tasks (jobs) [19]. Scheduling problems arise in a variety of situations, such as jobs in a manufacturing plant, or airplanes waiting for clearance to land or take off at an airport

The real problem of scheduling is how to determine the schedule that best meets the set objectives [18]. Scheduling objectives are usually measures of goodness of solution [19]. The objectives often vary from firm to firm. Some of the objectives of scheduling problems include minimising the total completion time, the number of tardy jobs, the maximum completion time, maximum lateness, etc [8].

Scheduling problems are combinatorial in nature [9]. As an example, for n jobs, m machines (n x m), the number of possible schedules is $(n!)^m$. This may look tractable when n and m have small values. However, as the values of n and m increase, the problem becomes complex. For example, there are 720 possible schedules for a 6 x 1 problem, while a 10 x 1 problem requires nearly 4 million schedules. A 20 x 1 problem solved on a computer that evaluates a million schedules per second would take nearly 80,000 years to evaluate all possible schedules. The difficulty of enumerating all the possible schedules in good time spurred the need to develop polynomial-time algorithms (heuristics methods) for solving scheduling problems. Heuristic methods are techniques for obtaining acceptable solutions to scheduling problems at a reasonable computational cost. While they do not guarantee optimal results, the techniques are relatively economical in terms of the computational resources used. Polynomial-time algorithms are algorithms whose number of computational steps does not grow exponentially with the input value.

In this work, the objective is to minimise the number of tardy jobs. A job is said to be tardy if it is completed after its due date. The number of tardy jobs is an important performance measure in scheduling, as it has great practical implications in an organization. When jobs are tardy, the goodwill of customers who own such jobs may be lost. This may have grave consequences for the organization. In some situations the lateness penalty may be large. It is therefore the desire of decision-makers to satisfy as many customers as possible by ensuring that jobs are completed on or before the due dates.

However, because of problem complexities (release and due dates constraints), it may be practically impossible to complete all jobs by their respective due dates at all times. So a schedule that yields lower values of the number of tardy jobs for any given problem is preferable.

The 'number of tardy jobs' criterion is particularly useful in organisations where the lateness penalty depends on *whether or not* a job is late, as against *how late* a job is. For example, if an aircraft is scheduled to land at a time after which it will have exhausted its fuel, then the results are just as catastrophic, whatever the scheduled landing time. So considering the number of tardy jobs as a performance measure is much more relevant in practical scheduling problems.

2. THE PROBLEM

Given the general one-machine scheduling problem, where a set J of n jobs has to be sequenced on a machine in order to minimize the number of tardy jobs, only one job can be processed at a time. The arrival time of every job J_i at the machine is known and denoted by r_i (release date). Each job J_i needs p_i time units on the machine (processing time), with a due date designated by d_i.

The time the processing of job J_i starts on the machine (start time) is designated as s_i with the property:

$$s_i \geq r_i \tag{1}$$

while its completion time (C_i) is defined as:

$$C_i = s_i + p_i \tag{2}$$

A job J_i is said to be tardy if it is completed after its due date ($C_i > d_i$).

We define:

$$U_i = \begin{cases} 1; if\, C_i > d_i \\ 0; otherwise \end{cases} \tag{3}$$

Then define, the number of tardy jobs (NT) as:

$$NT = \sum_{i=1}^{n} U_i \tag{4}$$

The symbolic representation of scheduling problems can be described by a three parameter notation: $\alpha \mid \beta \mid \gamma$ [10]

where:

α = machine environment (single machine, parallel machine (P), flow shop (F), job shop (J), etc)

β = Job characteristics (*prec*, *pmtn*, r_i , d_i etc.)

γ = Objective functions or performance measures

Using the above notations, the particular problem being considered is represented as

$$1 \mid r_i \mid \sum_{i=1}^{n} U_i$$

We assume that pre-emption is not allowed and that the problem is static and deterministic – i.e. the number of jobs, their processing times, due dates, and ready times are all known and fixed. These assumptions seem reasonable, as many real-life problems can be so modeled.

3. PREVIOUS WORK

The problem of minimizing the number of tardy jobs on a single machine with release dates ($1 \mid r_i \mid \sum_{i=1}^{n} U_i$) is NP-Hard in the strong sense [1]. Many special cases and/or relaxations of the problem have been studied by many researchers.

The first special case of the problem to be studied is problems for which all release dates are zeros. Moore [17] proposed an algorithm that solves the $1 \mid \mid \sum_{i=1}^{n} U_i$ problem in $O(nlogn)$ time. Later, Lawler and Moore [16] showed that, using dynamic programming, the $1 \mid \mid \sum_{i=1}^{n} w_i U_i$ problem is solvable in $O(n\Sigma)$ time. They were able to solve problems with up to 1,000 jobs. Potts and Wassenhove [21] considered the problem of scheduling n jobs, each having a processing time, a due date, and a weight, on a single machine to minimize the weighted number of late jobs. They proposed a branch and bound (B&B) algorithm that uses the linear programming lower bound. Also, computational results for problems with up to 1,000 jobs were given. Hallah and Bulfin [11] proposed a branch-and-bound algorithm that uses the bounds obtained from a surrogate knapsack, to solve the single machine weighted number of tardy jobs scheduling problem. They showed that instances with 2,500 jobs can be solved optimally in a reasonable amount of time. Phojanamongkolkij *et al*. [20] studied the single machine scheduling problem with a sequence-independent setup in which the weighted number of late jobs is minimized. A heuristic (called weighted forward algorithm) was proposed for the situation where jobs have zero release dates. Hatice and Fatih Tasgetiren [12] proposed a discrete particle swarm optimization algorithm and a traditional genetic algorithm to determine a sequence of n jobs to be processed through m machines, which minimises the number of tardy jobs.

The next set of problems studied by researchers is those problems with compatible release and due dates. Kise *et al*. [13] showed that the problem can be solved in $O(n^2)$ time provided that the release dates and due dates are compatible (that is, the jobs can be indexed so that $r_1 \leq r_2 \leq \ldots \leq r_n$ and $d_1 \leq d_2 \leq \ldots \leq d_n$). Lawler [15] proposed an $O(n^5)$ dynamic programming algorithm for the pre-emptive case ($1 \mid r_i, pmtn \mid \sum_{i=1}^{n} U_i$). Also, Carlier [2, 3] showed that, when the processing times are equal, the problem ($1 \mid p_i = p, r_i, \mid \sum_{i=1}^{n} U_i$) can be solved in $O(n^3 log(n))$ time.

Van den Akker and Hoogeveen [25] considered the single machine scheduling

problem of minimising the number of late jobs under stochastic situations. Four probability distributions on the processing times were considered. These are gamma, binomial, normal, and general (with the characteristic that all processing times are equally disturbed) distributions. Lambrechts *et al*. [14] proposed a Tabu Search procedure for generating robust project baseline schedules under stochastic resource availabilities. The Tabu Search uses a double neighborhood structure to allow for the generation of feasible project schedules that respect precedence, resource, and due dates constraints.

So far all the above literature addressed different special cases or relaxations of the general problem.

The general problem ($1 \mid r_i \mid \sum_{i=1}^{n} U_i$) requires that each job has distinct release and due dates. The literature reveals that not many researchers have addressed the general problem. This is probably due to the fact that the general problem is NP-Hard in the strong sense [1]. A number of researchers have considered the general problem, including Dauzere-Perez and Sevaux [5, 6, 7]. They have presented three mixed-integer linear programming formulations of the general problem, adopted a lagrangean relaxation approach for the weighted version of the problem ($1 \mid r_i \mid \sum_{i=1}^{n} w_i U_i$), and a branch and bound scheme for the same problem. Baptiste *et al*. [1] presented a branch and bound algorithm for the general problem. Sevaux and Dauzere-Peres [23] used a genetic algorithm to minimize the weighted number of late jobs on a single machine. Also, Sevaux and Thomin [24] developed two types of metaheuristic (simulated annealing and Tabu Search) for the parallel machine problem. They concluded that instances of up to 100 jobs can be solved efficiently. Sevaux and Sorensen [22] studied the single machine scheduling problem of minimizing the weighted number of late jobs ($1 \mid r_i \mid \sum_{i=1}^{n} w_i U_i$) and proposed a genetic algorithm to compute robust schedules when release dates are subject to small variations. To the best of our knowledge, only Dauzere-Perez [4] has presented a heuristic for the general problem (that in which jobs have different release and due dates). Since we are addressing this general problem in this paper, this heuristic has been selected for evaluation. Since the general problem has been proved to be NP-Hard in the strong sense [1], we propose three heuristics for the general problem.

4. SOLUTION METHODS

The problem of minimizing the number of tardy jobs with release dates on a single machine ($1 \mid r_i \mid \sum_{i=1}^{n} U_i$) is strongly NP-Hard [1]. Hence, polynomial-time approximation algorithm (such as heuristic) methods are needed to solve such

problems. Heuristics are methods of obtaining solutions to scheduling problems based on conjectures, insights, rule-of-thumb, etc. In this work, three heuristics are proposed for this problem.

4.1 Proposed heuristics

The three heuristics are now reviewed.

4.1.1 HR2 heuristic

The interval between the time when the job is released to the shop (release date) and the time the job is needed (due date) can be called the job's allowance. This is the maximum time a job can stay in the shop without being tardy. We propose to reduce the number of tardy jobs by scheduling the jobs according to an ascending order of the job allowance. This rule is called HR2 in this work. Scheduling jobs according to HR2 ensures that jobs with shorter allowances are scheduled in the earlier positions, while jobs with longer allowances are scheduled later. The steps for the HR2 heuristic are outlined as:

Step 1: Compute HR2 (job allowance) index by $HR2_i = d_i - r_i$
Step 2: Form a list *L* by arranging jobs according to the ascending order of the HR2 index computed in Step 1

Step 3: Schedule jobs according to list *L* which was obtained from Step 2

The HR2 heuristic is easy to apply.

4.1.2 HR3 heuristic

A careful examination of equations (1) to (4) reveals that the number of tardy jobs is indeed a function of three given parameters – namely, p_i, r_i, d_i. So it was proposed that scheduling jobs according to the ascending order of the sum of the above three parameters would minimize the number of tardy jobs. The HR3 heuristic ensures that earlier released, due, and short jobs are scheduled in the earlier positions. The steps for the HR3 heuristic are now outlined:

Step 1: Compute HR3 index by $HR3_i = p_i + r_i + d_i$

Step 2: Form a list *L* by arranging jobs according to the ascending order of the HR3 index computed in Step 1

Step 3: Schedule jobs according to list *L* which was obtained from Step 2

The HR3 heuristic is also easy to apply.

4.1.3 EOO heuristic

One of the assumptions made in this paper is that the scheduling problem being considered is static and deterministic. It is static in the sense that, although the jobs have release dates, these dates are known and fixed. Because the processing and due dates are also known with certainty, the problem is deterministic. The third

proposed heuristic (called EOO) makes use of this vital information. If jobs are not to be tardy, priority must be given to the due dates of the jobs. What the EOO does is to consider the next job that is due (out of all the given jobs, but still to be scheduled) and checks if scheduling the job at that time makes the job tardy. If the job is likely to be tardy, the EOO will not schedule the job but rather consider the next job that is due while adding the previous job to the list of tardy jobs. If the job is unlikely to be tardy, the EOO will schedule the job before considering other jobs. The key is that if a job is tardy at time t_1, the job might as well be scheduled at any other time t_2 (where $t_2 > t_1$). The EOO heuristic is now outlined:

Step 1: Initialization

Job_Set1 = $\{Ji\}_{i=1}^{n}$ 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 selected 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.

4.2 Selected solution methods

Since to the best of our knowledge only Dauzere-Perez [4] has presented a heuristic for the general problem (problems in which jobs have different release dates), this heuristic has been selected for evaluation.

4.2.1 Dauzere (DAU) heuristic

The steps of the DAU heuristic are now presented.

Dauzere-Perez Stephane (DAU) heuristic

Step 0: Initialise

$t = \min_{J_i \in J} r_i$ (i.e. the minimum ready time of all jobs)

I = {.} (I is the set of early or sequenced jobs, set it to an empty set)

I_t = {.} (I_t is the set of tardy jobs, set it to an empty set)

$\bar{I}$ = $\{Ji\}_{i=1}^{n}$ ($\bar{I}$ is the set of jobs yet to be sequenced)

Step 1: Choose the job with the smallest due date among the jobs that have arrived at time t from $\bar{I}$

Step 2: Add the job chosen in Step 1 to I and remove the same job from $\bar{I}$. Compute start time $S_i = t$ and completion time $C_i = S_i + p_i$

Step 3: If this job is late (i.e. $C_i > d_i$), proceed to Step 4; otherwise jump to Step 6

Step 4: Remove this job from I and add it to I_t

Step 5: Update the start and completion time as: $S_i = t$ and $C_i = S_i$

Step 6: Compute new time as: $t = max\ (\ C_i,\ \min_{J_i \in \bar{I}} r_i\ )$

Step 7: If set $\bar{I}$ is not empty, go back to Step 1; otherwise proceed to Step 8

Step 8: Form the sequence of jobs by appending jobs in set I with jobs in I_t without changing the sequence. The number of tardy jobs is the number of jobs in set I_t

Step 9: Stop.

5. DATA ANALYSIS

A software package was developed in Microsoft Visual Basic 6.0 to generate random single machine problems. A total of 50 problems for 22 different problem sizes (ranging from 3 to 500 jobs) was randomly generated. A total of 1,100 (22*50) single machine problems was solved.

The processing time of jobs was randomly generated with values ranging between 1 and 100 inclusive. The ready time of jobs was also randomly generated with values ranging between 0 and $\sum_{i=1}^{n} P_i$ inclusive, while the due date was also randomly generated with values ranging between ($r_i + p_i$) and ($r_i + 2*p_i$) inclusive.

All the above solution methods were included in the program that was developed. When the program was run, the value of the number of tardy jobs obtained for each solution method and each problem size was computed and saved in a data file, which was then exported to Statistical Analysis System (SAS) version 9.1 for detailed analysis. (SAS is a versatile statistical package that enables credible conclusions to be drawn from the results.) The general linear models (GLM) in SAS were used to compute the mean of the number of tardy jobs for each of the solution methods and each of the problem sizes. The GLM procedure in SAS was also used to carry out test of means (means separation) so as to determine whether the differences observed in the mean value of number of tardy jobs obtained by various solution methods are significant or just due to chance.

6. RESULTS

The results obtained when the mean value of number of tardy jobs was computed for each solution method and problem size are shown in Table 1. The EOO heuristic gave the lowest number of tardy jobs for all the problem sizes considered, indicating a better performance when compared with other methods. This was

followed by the DAU method. The HR3 method came third in ranking for 3 to 120 jobs problems, while for 140 to 500 jobs problems the HR2 method came third. Therefore, based on the minimum mean value of the 'number of tardy jobs' criterion, the EOO heuristic outperformed all the other methods (DAU, HR2 and HR3).

Problem	Mean of number of tardy jobs			
Size	EOO	DAU	HR2	HR3
3x1	0.6	0.94	0.76	0.74
4x1	1.38	1.68	2.02	1.9
5x1	1.58	2.44	2.54	2.42
6x1	2.3	2.94	3.48	3.28
7x1	2.74	3.62	4.8	4.52
8x1	3.46	4.6	5.74	5.46
9x1	4.18	5.16	6.72	6.16
10x1	4.76	5.74	7.28	6.94
12x1	5.92	6.82	9.58	9
15x1	8.2	9.48	12.42	11.98
20x1	11.5	12.2	17.82	16.9
25x1	14.5	15.58	22.26	21.84
30x1	17.36	18.72	27	26.22
40x1	23.8	25.9	37.02	36.44
50x1	31.02	32.64	47.32	46.78
100x1	62.86	63.7	97	96.72
120x1	75.88	77.76	117.02	116.82
140x1	87.64	89.04	136.92	136.94
200x1	125.26	126.68	196.4	197.08
300x1	188.88	190.16	296.48	296.84
400x1	251.4	252.44	296.16	397.04
500x1	314.36	315.18	496.7	497.16

Sample size = 50

Table 1: Mean of number of tardy jobs obtained from the heuristics by problem size

EOO, HR2, and HR3 methods were compared with the DAU method using the approximation ratio (this is the ratio of the value of the number of tardy jobs obtained using a solution method to that obtained using DAU method); the results obtained are shown in Figure 1. It was observed that the EOO method performs better than the DAU method.

Figure 2 shows the mean time taken (in seconds) by the EOO, DAU, HR2, and HR3 methods for 3 to 500 jobs. It is interesting to observe that the EOO heuristic is faster than the other three when the number of jobs exceeds 180 (Figure 2). For example, while the EOO heuristic required about 0.33 seconds to solve a problem involving 500 jobs, the DAU heuristic took about 0.43 seconds to solve the same problem

(Figure 2). However, when the number of jobs is between 50 and 180, the EOO heuristic is slower than the others. There is no solution method that is consistently faster when the number of jobs is less than 50 (Figure 2).

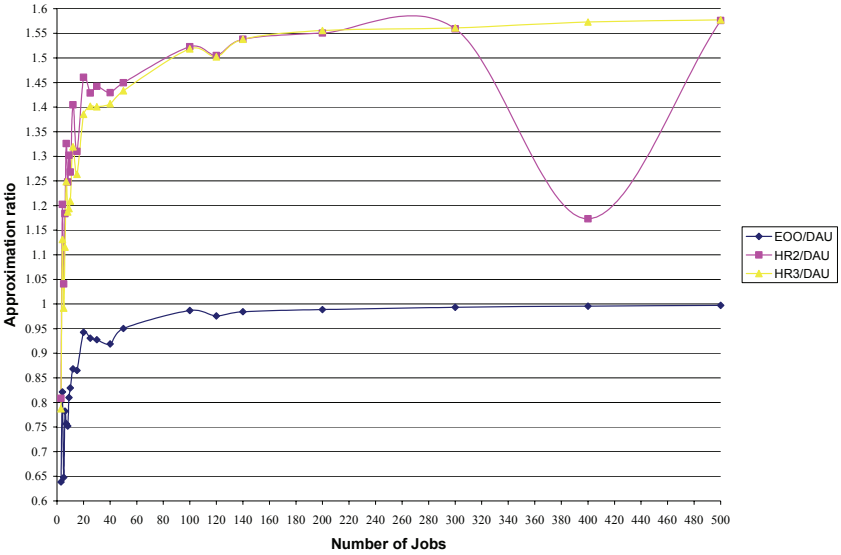

Figure 1: Approximation ratio of EOO, HR2, and HR3 compared with DAU for 3 to 500 jobs

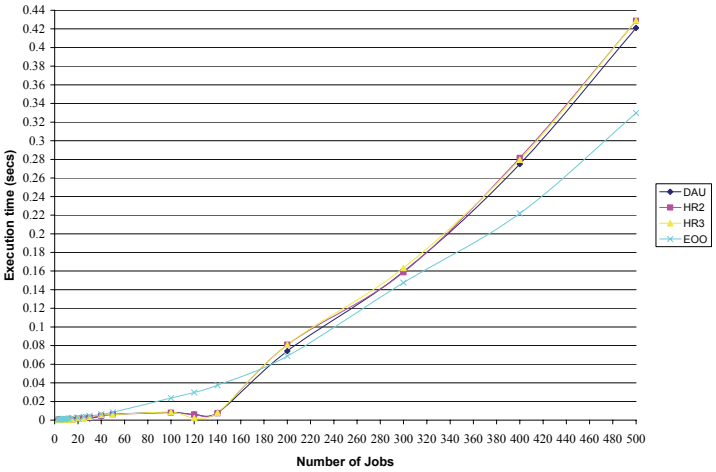

Figure 2: Time taken by EOO, DAU, HR2, and HR3 methods for 3 to 500 jobs

The mean number of tardy jobs and the mean of execution time by each solution method was subjected to statistical tests. The results are summarized in Table 2, where the solution methods in both the effectiveness and efficiency columns have been ordered in ascending effectiveness (value of the number of tardy jobs) and efficiency (execution time) for each problem size. The superscript on some solution methods indicates that there is a significant difference in performance between that particular solution method and the solution method(s) specified in the column(s) being referred to by the letter(s) in the superscripts. For example, the mean value of the number of tardy jobs given by EOO (effectiveness column) is significantly different from (better than) that of DAU, HR2, and HR3 ($P \leq 0.05$) for $3 \leq n \leq 200$ problems, meaning that the quality of solution given by the EOO heuristic is much better than that of the DAU, HR2, and HR3 methods when the number of jobs is less than or equal to 200 (Table 2). Under the same problem loading and at

Problem Size	Effectiveness				Efficiency			
	A	B	C		D	E	E	
3x1	EOO^{*A,B,C},	DAU,	HR3,	HR2	DAU,	HR3,	HR2,	EOO
4x1	EOO^{*A,B,C},	DAU,	HR3,	HR2	DAU,	HR3,	HR2,	EOO
5x1	EOO^{*A,B,C},	DAU,	HR3,	HR2	DAU,	HR3,	HR2,	EOO
6x1	EOO^{*A,B,C},	DAU,	HR3,	HR2	DAU,	HR3,	HR2,	EOO
7x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
8x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
9x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
10x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
12x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
15x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
20x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
25x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
30x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
40x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
50x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
100x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
120x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
140x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
200x1	EOO^{*A,B,C},	DAU^{*B,C},	HR3,	HR2	DAU,	HR3,	HR2,	EOO
300x1	EOO^{*B,C},	DAU^{*B,C},	HR3,	HR2	EOO,	HR2,	DAU,	HR3
400x1	EOO^{*B,C},	DAU^{*B,C},	HR3,	HR2	EOO,	DAU,	HR3,	HR2
500x1	EOO^{*B,C},	DAU^{*B,C},	HR3,	HR2	EOO,	DAU,	HR2,	HR3

Sample size=50

Note: *A,B,C indicates significant result at 5% level from solution methods in columns A, B and C. *B, C indicates significant result at 5% level from solution methods in columns B and C

Table 2: Performance with respect to effectiveness and efficiency

the same significant level, the mean value of the number of tardy jobs given by DAU is significantly different from (better than) that of HR3 and HR2. For $300 \leq n \leq 500$ problems, there are no significant differences in the performance (effectiveness) of all the solution methods at $P \leq 0.05$ (Table 2).

When the mean of the time taken was also subjected to statistical test, there was no significant difference in the time taken by EOO, DAU, HR2, and HR3 ($P \leq 0.05$) for $3 \leq n \leq 500$ problems (see efficiency column of Table 2).

7. CONCLUSION

In this paper, the scheduling problem of minimizing the number of tardy jobs with release dates on a single machine is considered. Three heuristics – EOO, HR2, and HR3 – were proposed for solving this problem, and were then compared with a heuristic (DAU) selected from the literature. The solution methods were all evaluated and tested with respect to both effectiveness (closeness of value of number of tardy jobs to the optimal) and efficiency (how fast a solution can be obtained).

Based on effectiveness, one of the proposed heuristics, EOO, outperformed the others for $3 \leq n \leq 500$ problems. Also, the EOO is as fast as the DAU, HR2, and HR3 methods when the number of jobs is less than 50; slower than the others for 50 to 180 jobs; and faster than the others when the number of jobs exceeds 180. The EOO heuristic obtains quality solutions that are both effective and efficient. The EOO heuristic is, therefore, recommended for the single machine scheduling problem of minimizing the number of tardy jobs with release dates.

8. REFERENCES

[1] **Baptiste, P., Laurent, P. and Eric, P.** 2003. A branch and bound to minimize the number of late jobs on a single machine with release time constraints, *European Journal of Operational Research,* 144(1), pp. 1-11.

[2] **Carlier, J.** 1981. Probleme a une machine et algorithms polynomiaux, *Questio* 5.

[3] **Carlier, J.** 1982. The one-machine sequencing problem, *European Journal of Operational Research,* 11(1), pp. 42-47.

[4] **Dauzere-Perez, S.** 1995. Minimizing late jobs in the general one-machine scheduling problem, *European Journal of Operational Research,* 81(1), pp. 134-142.

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

[6] **Dauzere-Perez, S. and Sevaux, M.** 2003. Using lagrangean relaxation to minimize the weighted number of late jobs, *Naval Research Logistics,* 50, pp. 273-288.

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

[8] **Edwin, C.T.C., Milhail, Y.K. and Alexander, V.T.** 1996. Single machine group scheduling with two ordered criteria, *Journal of Operational Research,* 47, pp. 315-320.

[9] **French, S.** 1982. *Sequencing and scheduling.* Ellis Horwood Limited.

[10] **Graham, R.L., Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G.** 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics,* 5, pp. 287-326.

[11] **Hallah, R.M. and Bulfin, R. L.** 2003. Minimizing the weighted number of tardy jobs on a single machine, *European Journal of Operational Research,* 145(1), pp. 45-56.

[12] **Hatice, Ucar and Fatih Tasgetiren, M.** 2006. A particle swarm optimization algorithm for permutation flow shop sequencing problem with the number of tardy jobs criterion, IMS 2006: 5th International Symposium on intelligent manufacturing systems: Agents and virtual worlds, May 29-31, 2006, Sakarya, Turkey.

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

[14] **Lambrechts, O., Demeulemeester, E.L. and Herroelen, W.S.** 2007. Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities, *Journal of Scheduling (in press).*

[15] **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,* 26, pp. 125-133.

[16] **Lawler, E.L. and Moore, J.M.** 1969. A functional equation and its application to resource allocation and sequencing problems, *Management Science,* 16, pp. 77-84.

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

[18] **Oluleye, A.E. and Oyetunji, E.O.** 1999. Performance of some static flowshop scheduling heuristics, *Directions in Mathematics,* pp. 315-327.

[19] **Uthaisombut, P.** 2000. New directions in machine scheduling. PhD thesis, Michigan State University.

[20] **Phojanamongkolkij, N., Grabenstetter, D. and Ghrayeb, O.** 2003. Scheduling jobs to improve weighted on-time performance of a single machine, *Journal of Manufacturing Systems,* 22, pp. 148-156.

[21] **Potts. C.N. and Wassenhove, L. N.** 1988. Algorithms for scheduling a single machine to minimize the weighted number of late jobs, *Management Science*, 34, pp. 843–858.

[22] **Sevaux, M. and Sorensen, K.** 2004. A genetic algorithm for robust schedules in a just-in-time environment with ready times and due dates, *4OR – Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2, pp. 129-147.

[23] **Sevaux, M. and Dauzere-Perez, S.** 2003. Genetic algorithms to minimize the weighted number of late jobs on a single machine, *European Journal of Operational Research*, 151(2), pp. 296-306.

[24] **Sevaux, M. and Thomin, P.** 2001. Heuristics and metaheuristics for parallel machine scheduling: A computational evaluation, in *Proceedings of 4th Metaheuristics International Conference, MIC 2001*, pp. 411-415, Porto, Portugal, 16-20 July 2001.

[25] **Van den Akker, J.M. and Hoogeveen, J.A.** 2004. Minimizing the number of tardy jobs in cases of stochastic processing times with minimum success probabilities, *Institute of Information and Computing Sciences, Utrecht University Technical Report UU-CS-2004-067*.