http://dx.doi.org/10.7166/26-3-1030

A HEURISTIC APPROACH TO MINIMISING MAXIMUM LATENESS ON A SINGLE MACHINE

B. Çalış[1]*, S. Bulkan[2] & F. Tunçer[3]

[1,2]Department of Industrial Engineering
Marmara University, Turkey
[1]bcalis@marmara.edu.tr, [2]sbulkan@marmara.edu.tr

[3]Department of Computer Engineering
Marmara University, Turkey
ferittuncer@marun.edu.tr

ABSTRACT

This paper focuses on the problem of scheduling on a single machine to minimise the maximum lateness when each job has a different ready time, processing time, and due date. A simple procedure is developed to find a better solution than the early due date (EDD) algorithm. The new algorithm suggested in this paper is called Least Slack Time - Look Ahead (LST-LA), which minimises the maximum lateness problem. Computational results show that when the number of jobs increases, LST-LA outperforms EDD.

OPSOMMING

Die artikel konsentreer op die skedulering van ‘n enkele masjien om die maksimum laatwees probleem, wanneer elke taak ‘n verskillende gereedheidstyd, prosesseertyd en keerdatum het, te minimeer. ‘n Eenvoudige prosedure om ‘n beter oplossing tot die vroeë keerdatum algoritme te bepaal, is ontwikkel. Die nuwe algoritme word die “Least Slack Time Look Ahead” genoem en dit minimiseer die maksimum laatwees probleem. Simulasie resultate toon dat soos die aantal take toeneem, vertoon die nuwe algoritme beter as die vroeë keerdatum algoritme.

[1] The author was enrolled for an PhD degree in the Department of Industrial Engineering, Marmara University, Turkey

[3] The author was enrolled for a BSc degree in the Department of Computer Engineering, Marmara University, Turkey

* Corresponding author

1 INTRODUCTION

This paper considers the problem of finding an effective schedule for n jobs with different release dates on a single machine, in order to minimise the maximum lateness. The single machine scheduling problem is one of the important problem types in machine scheduling models; it can be used to solve the single machine models, or it gives an initial solution to the decomposition of big problems into sub-problems, such as job shops or flow shops. Minimising maximum lateness (L_{max}) is an important objective if all preceding activities must be completed before the rest can begin in a project; i.e., a very late activity may cause a delay in the project. In addition, L_{max} may be used as an aid for solving other problems [1].

To define a scheduling problem, the well-known three-field notation of Graham et al. [2], $\alpha|\beta|\gamma$, is used, where α, β, and γ represent the machine environment, job characteristics (problem constraints), and objective function respectively. The scheduling problem studied in this work is minimising maximum lateness (L_{max}) for n jobs, with release dates (r_j) on a single machine. The problem is denoted by $1|r_j|L_{max}$ and shown as NP-hard by Lenstra et al. [3]. When the difference of C_j-d_j (completion time of job j – due date of job j) is positive, the job is said to be late; if the difference of C_j-d_j is negative, the job is said to be early; if the result of the difference equals zero, the job is said to be on time.

This paper provides a brief and up-to-date literature review of the single machine maximum lateness with release times. It then proposes a simple and effective heuristic called Least Slack Time - Look Ahead (LST-LA) to solve the $1 \mid r_j \mid L_{max}$ problem. The solutions obtained by the full enumeration (for the problems up to 13 jobs) and early due date (EDD) rule (for the problems with more than 13 jobs) are compared. Experimential results of the study show that the proposed algorithm (LST-LA) outperforms the EDD algorithm, on average, by at least 400 per cent on the Carlier's instances.

The outline of this paper is as follows. In Section 2, the maximum lateness problem is defined. In Section 3, the analysis of the proposed heuristic algorithm, procedures, and a computational example are explained in detail. Section 4 presents computational results based on the randomly generated problem instances for the single machine minimising maximum lateness problem under study. Section 5 presents the statistical analysis of the results. Test results on Carlier's problems are also presented in Section 5.2. Finally, in Section 6, overall conclusions are drawn and future research paths are highlighted.

2 MINIMISING THE MAXIMUM LATENESS PROBLEM

The problem without release times ($1||L_{max}$) is optimally solvable by the EDD first in polynomial time [4]. The EDD rule can be defined as the set of n jobs, with known processing times and due dates; the minimum value of L_{max} is achieved by sequencing the jobs in non-decreasing order of their due dates [5]. A detailed literature review on the single machine maximum lateness problem with release times ($1|r_j|L_{max}$) can be found in Sels and Vanhoucke [6]. The well-studied problem $1|r_j|L_{max}$ is a special case of problem $1|prec;r_j|L_{max}$.

Oyetunji and Oluleye [7] proposed a heuristic to reduce the number of tardy jobs by scheduling the jobs according to an ascending order of the job allowance; test results show that the given algorithm is faster than others when the number of jobs is large.

McMahon and Florian [8] provide efficient branch and bound algorithms to solve this problem. Lagaweg et al. [9] studied scheduling jobs on a single machine subject to given release dates and precedence constraints; they described applications to the theory of job-shop scheduling and to a practical scheduling situation.

Frederickson [10] showed that the problem of scheduling n unit-time tasks with integer release times and deadlines is solvable in O(n log n) time, if a sufficient amount of uninitialised space is available. Baker et al. [11] considered that n jobs are to be processed on a single machine, subject to release dates and precedence constraints, and they presented an O(n2) algorithm for this problem.

Gordon [12] considered the optimal assignment of slack due-dates and sequencing in the single-machine shop to the case when pre-emption is allowed and there are precedence constraints and ready times of jobs. The study shows that under special conditions, the presented algorithm may be used when pre-emption is not allowed.

Oyetunji and Oluleye [13] considered the single machine scheduling problem subject to release dateto minimise total completion time and the number of tardy jobs and a proposed a solution algorithm, which was recommended when there are 30 or more jobs in the problem.

Monma and Potts [14] studied the single machine scheduling problem with sequence-dependent family setup times. Results of their study showed that "for the maximum lateness problem, there is an optimal schedule where the jobs within each batch are ordered by the EDD rule".

Schrage [15] proposed an algorithm that considers scheduling an available job with the largest tail time. Carlier [16] studied how to solve the $1|r_j, q_j|L_{max}$ problem. Results from Carlier's algorithms shows that this is the most promising approach in the literature.

A lot of researchers considered minimising the maximum lateness problem under different constraints. Due to the NP-hard nature of the problem, a number of different heuristics have been developed. In this study, minimising the maximum lateness problem is studied under a release date constraint; and a simple and efficient heuristic algorithm called LST-LA is proposed and tested on randomly-generated problems.

The next section includes detailed information about the proposed heuristic algorithm.

3 PROPOSED LST-LA ALGORITHM

The problem of $1|r_j|L_{max}$ is strongly NP-hard [15]. The problem considers n independent jobs (j=1,2,…,n) with unequal release dates (r_j) on a single machine to minimise maximum lateness. There are no precedence constraints between jobs, and each job has positive due dates, d_j ($d_j \geq 0$). The machine is continuously available and can process one job at a time.

The problem of $1||L_{max}$ is the best-known special case of $1|prec|h_{max}$. The function h_j is then defined as C_j-d_j, and the algorithm results in the schedule that orders the jobs in increasing order of their due date [14]; in order to minimise the maximum lateness the algorithm processes the jobs in non-decreasing due date order. This dispatching rule is known as EDD, or Jackson's rule, after Jackson [16] who studied it in 1955 [17].

The results obtained by LST-LA are compared with the results from the EDD rule. The notations below are used to describe the scheduling problem.

Problem procedures: the flow chart of the LST-LA algorithm is given in Figure 1.

Notations:

J : set of scheduled operations
J^d : candidate to schedule jobs at time t, ($r_j \leq t$)
J^c : set of unscheduled operations
t : scheduling time
L_{max} : maximum lateness

L_j : lateness of job j
P_j : processing time of job j
d_j : due date of job j
C_j : completion time of job j

The machine is assumed to be continuously available and can process, at most, one job at a time.

The jobs may not be pre-empted, and each job j is characterised by its processing time p_j, its release time r_j, and its due date d_j.

3.1 Algorithm definition

The flow chart of the proposed LST-LA algorithm can be seen in Figure 1.

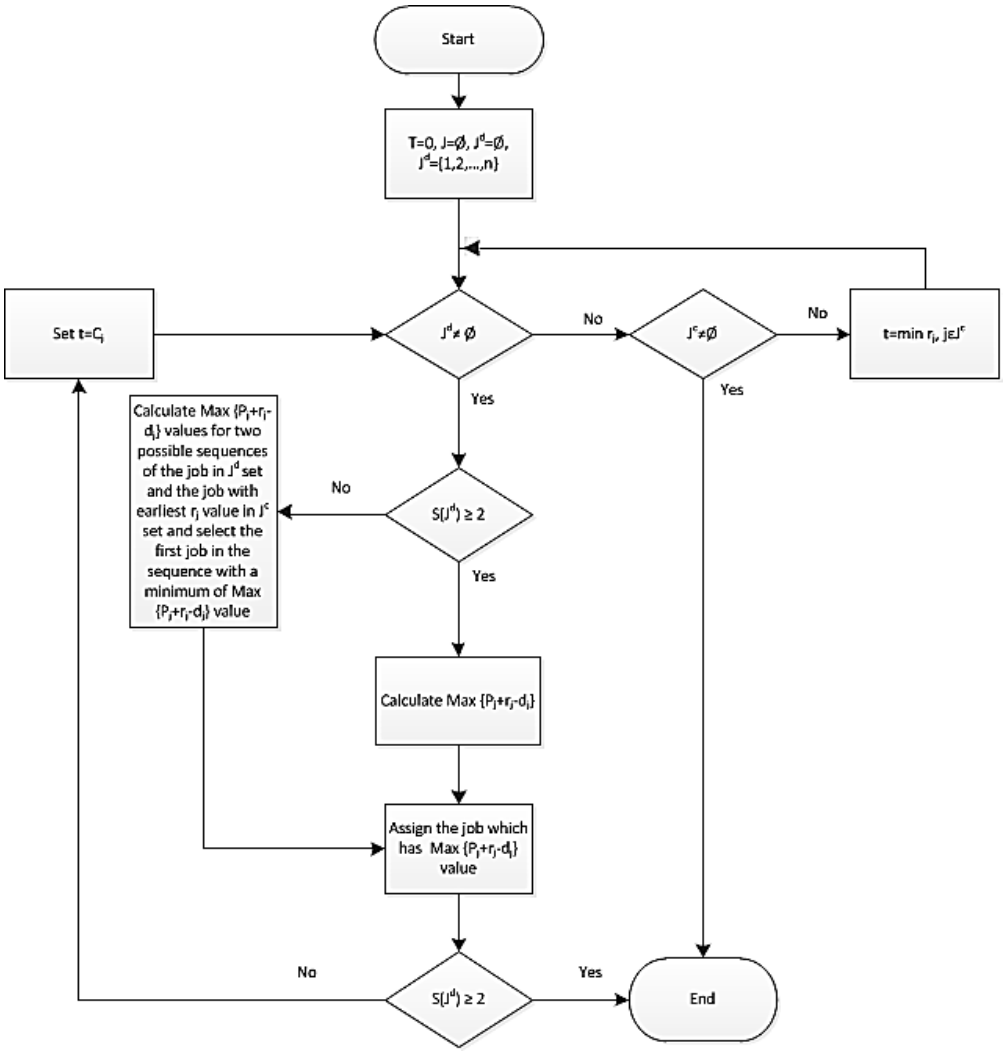

Figure 1: Flow chart of the LST-LA algorithm

The pseudo code of the proposed model is given in Figure 2.

```
01 t=0, J = new empty set, J^c = job input;
02 for each job in J^c
03        if time >= job.releaseDate
04                add this job to J^d;
05                if J^d > 1
06                        schedule the job with max lateness available in J^c;
07                        update time, J^c, J;
08                        if J^c > 0
09                                goto line 3
10                        else
11                                EXIT
12                if J^d = 1
13                        mainJob = the job in J^d
14                        if there is a job in J^c else then this candidate
15                                alternativeJob = next first available job in J^c;
16                                If scheduling mainJob before alternativeJob result in a
lower
                                        Lmax than scheduling alternative job before main
job
17                                      schedule mainJob
18                                      if J^c > 0
19                                              goto line 3
20                                      else
21                                              EXIT
22                                else
23                                      schedule alternativeJob
24                                if J^c > 0
25                                      goto line 3
26                                else
27                                      EXIT
28                        else
29                        schedule main job
30                        if J^c > 0
31                                goto line 3
32                        else
33                                EXIT
34        else
35                time = next available jobs release date;
36                goto line 3
```

Figure 2: Pseudo code of LST-LA algorithm

3.2 A computational example

In this part of the study, execution of the algorithm is presented using a small example in order to demonstrate the algorithm. Sample data is given in Table 1.

Table 1: Sample data of execution

Jobs	1	2	3	4	5	6	7	8
P_j	4	9	3	3	6	8	8	12
r_j	0	12	0	20	2	22	15	30
d_j	10	29	8	30	16	33	42	48

Step1. t=0 J= ϕ, J^d = ϕ, L_{max}=-∞ and J^c={1,2,3,4,5,6,7,8}
Step2. J^d = {1,3} since they are ready at time 0.
Max{P_j+r_j – d_j} ={P_1+r_1- d_1, P_3+r_3-d_3}
= {4+0-10, 3+0-8}
={-6,-5}
=-5
J={3}, J^c={1,2,4,5,6,7,8}, C_3 =3, t=3, L_3=-5 < 0 and since L_3 > L_{max}, L_{max}=-5

Step3. J^c is not an empty set ($J^c \neq \phi$) so go to Step 2.
Step2. t=3
J^d = {1,5} and t=3 > r_1 =0 so r_1 =3 and t=3 > r_5 =2 so r_5 =3
Max{$P_j+r_j - d_j$} ={$P_1+r_1- d_1$, $P_5+r_5-d_5$}
= {4+3-10, 6+3-16}
={-3,-7}
=-3
J={3,1}, J^c={2,4,5,6,7,8}, C_1 =7, t=7, L_1=-3 < 0 and since $L_1 > L_{max}$, L_{max}=-3

Step3. J^c is not an empty set ($J^c \neq \phi$), so go to Step 2.

Step2. t=7 and J^d set has only one job (J_5), check the next available job (j_2 and r_2 =12), which has next minimum r_j value in J^c set and calculate L_{max} values for two possible sequences of two jobs, and then schedule the first job in the sequence that results in minimum L_j value for the second job in the sequence.
J^d = {2, 5}

For the sequence of 5, 2:
Since t=7 > r_5 =2 so r_5 =7

L_5={$P_5+r_5-d_5$}= {6+7-16}=-3 and C_5=13, t=13 > r_2 =12 so r_2 =13.
L_2={$P_2+r_2- d_2$}= {9+13-29}=-7 and C_2=22, t=22

Max {L_5, L_2} = -3 < 0 so L_{max} =-3 (for the sequence of 5, 2)

For the sequence of 2, 5:
Since t=7 < r_2 =12 so keep the machine idle until time 12 and set t=12

L_2={$P_2+r_2-d_2$}= {9+12-29}=-8 and C_2=21, t=21 > r_5 =2 so r_5 =21
L_5={$P_5+r_5-d_5$}= {6+21-16}=11 and C_5=27, t=27

Max {L_2, L_5} =11 > 0 so L_{max} =11 (for the sequence of 2, 5)

Since L_{max} value is minimum for the first sequence (i.e., sequence of 5, 2) Job 5 will be selected and assigned to the J set. So, J={3,1,5}, J^c={2,4,6,7,8}, C_j =13, t=13, L_j=-3 < 0 L_{max}=-3.

Step3. J^c is not an empty set ($J^c \neq \phi$), so go to Step 2.

At the end of the solution all sets are calculated as:
J={3,1,5,2,6,4,7,8}, J^c= ϕ, C_8 =53, t=53, L_3 =-5, L_1=-3, L_5=-3, L_2=-7, L_6=-3, L_4=3, L_7=-1 and L_8=5. LST-LA yields an L_{max}=5, which is identical to the L_{max} obtained by EDD solution although the sequence is different.

4 TEST PROBLEMS AND RESULTS

First, the problem data is generated as follows:

p_j is generated from a discrete uniform distribution between 5 to 50
r_j is generated from a discrete uniform distribution between 0 and $\sum_{j=1}^{n} p_j$
d_j is generated from a discrete uniform distribution between 0 and [max r_j + max p_j]

The problem is then tested with 15 independent samples with a constant size of 10. LST-LA, EDD, and full enumeration (FE) solutions are calculated and compared in Table 2.

Table 2: Comparison table for set size 10

Job size	Problem set	L_{max}-FE	L_{max}-EDD	L_{max}-LST-LA	%difference btw EDD & FE	%difference btw LST-LA & FE	%difference btw LST-LA & EDD
10	1	**257**	332	260	29,18%	1,17%	-27,69%
	2	**163**	203	**163**	24,54%	0,00%	-24,54%
	3	**127**	170	**127**	33,86%	0,00%	-33,86%
	4	**245**	299	**245**	22,04%	0,00%	-22,04%
	5	**275**	303	**275**	10,18%	0,00%	-10,18%
	6	**131**	206	144	57,25%	9,92%	-43,06%
	7	**226**	320	269	41,59%	19,03%	-18,96%
	8	**191**	209	**191**	9,42%	0,00%	-9,42%
	9	**146**	214	161	46,58%	10,27%	-32,92%
	10	**273**	308	280	12,82%	2,56%	-10,00%
	11	**200**	202	**200**	1,00%	0,00%	-1,00%
	12	**189**	225	**189**	19,05%	0,00%	-19,05%
	13	**115**	142	**115**	23,48%	0,00%	-23,48%
	14	**175**	204	208	16,57%	18,86%	1,92%
	15	**113**	**113**	130	0,00%	15,04%	13,08%
Bold values in the column of Lmax-EDD and Lmax-LST-LA are optimal solutions.					**23,17%**	**5,12%**	-17,41%

LST-LA obtained an optimal solution eight times out of 15 for randomly-selected problems, the EDD rule obtained an optimal solution only once, and the EDD rule ended up with a better solution than the LST-LA algorithm only once. The percentage differences show that the EDD rule yielded a maximum lateness up to 57 per cent away from the optimum value obtained by the full enumeration, whereas the LST-LA algorithm yielded a maximum lateness up to 19.3 per cent away from the optimum solution. The average percentage differences are calculated to be 23.17 per cent between EDD and FE, and 5.12 per cent between LST-LA and FE. LST-LA yields solutions that are 17.41 per cent lower than EDD, and these results mean that LST-LA gives a better solution than EDD, on average.

Hereafter, the problem set size is increased to 30, 50, 80, 100, and 150 jobs, and the results are listed in Table 3, Table 4, Table 5, Table 6, and Table 7, respectively.

When the job size increases to 30 jobs (Table 3), LST-LA yields solutions that are 8.16 per cent lower than EDD, on average. The LST-LA algorithm obtains better results than EDD in 13 out of the 15 test problems, whereas EDD yields better results than LST-LA only twice out of the 15 test problems. The LST-LA algorithm yields better results than EDD 8.16 per cent of the time, on average.

Table 3: Comparison table for set size 30

Job size	Problem set	L_{max}-EDD	L_{max}-LST-LA	% difference btw LST-LA-EDD
30	1	619	**587**	-5,17%
	2	695	**660**	-5,04%
	3	**828**	852	2,90%
	4	858	**670**	-21,91%
	5	729	**684**	-6,17%
	6	713	**637**	-10,66%

Table 3 (cont.): Comparison table for set size 30

Job size	Problem set	L_{max}-EDD	L_{max}-LST-LA	% difference btw LST-LA-EDD
	7	698	**627**	-10,17%
	8	**749**	751	0,27%
	9	678	**592**	-12,68%
	10	811	**681**	-16,03%
	11	908	**871**	-4,07%
	12	792	**705**	-10,98%
	13	683	**572**	-16,25%
	14	683	**667**	-2,34%
	15	777	**745**	-4,12%
				-8,16%

The LST-LA algorithm yields solutions that are 4.61 per cent lower than EDD, on average, for the problems with 50 jobs, as seen in Table 4.

Table 4: Comparison table for set size 50

Job size	Problem set	L_{max}-EDD	L_{max}-LST-LA	%difference btw LST-LA-EDD
50	1	1267	**1223**	-3,47%
	2	1001	**980**	-2,10%
	3	1566	**1380**	-11,88%
	4	1256	**1247**	-0,72%
	5	1333	**1219**	-8,55%
	6	1605	**1513**	-5,73%
	7	1216	**1031**	-15,21%
	8	1272	**1265**	-0,55%
	9	945	**936**	-0,95%
	10	1284	**1324**	3,12%
				-4,61%

The LST-LA algorithm yields solutions that are 8.46 per cent lower than EDD, on average, for the problems with 80 jobs, as seen in Table 5.

Table 5: Comparison table for set size 80

Job size	Problem set	L_{max}-EDD	L_{max}-LST-LA	%difference btw LST-LA-EDD
80	1	2379	**2179**	-8,41%
	2	2310	**2239**	-3,07%
	3	2275	**1979**	-13,01%
	4	2260	**2102**	-6,99%
	5	2277	**2015**	-11,51%
	6	1898	**1729**	-8,90%
	7	1694	**1612**	-4,84%
	8	1984	**1846**	-6,96%
	9	2054	**1883**	-8,33%
	10	2280	**1993**	-12,59%
				-8,46%

For 100 job problems, LST-LA yields solutions that are 6.65 per cent lower than EDD, on average, as seen in Table 6.

Table 6: Comparison table for set size 100

Job size	Problem set	L_{max}-EDD	L_{max}-LST-LA	%difference btw LST-LA - EDD
100	1	2479	**2334**	-5,85%
	2	2944	**2604**	-11,55%
	3	2836	**2642**	-6,84%
	4	2994	**2623**	-12,39%
	5	2416	**2394**	-0,91%
	6	2545	**2442**	-4,05%
	7	2617	**2312**	-11,65%
	8	2599	**2504**	-3,66%
	9	2476	**2391**	-3,43%
	10	2491	**2337**	-6,18%
				-6,65%

For the problems with 150 jobs, as can be seen in Table 7, improvement of LST-LA's results is 5.44 per cent compared with EDD, on average.

Table 7: Comparison table for set size 150

Job size	Problem set	L_{max}-EDD	L_{max}-LST-LA	%difference btw LST-LA -EDD
150	1	3501	**3409**	-2,63%
	2	3689	**3528**	-4,36%
	3	3484	**3428**	-1,61%
	4	3784	**3721**	-1,66%
	5	4207	**3752**	-10,82%
	6	4461	**4033**	-9,59%
	7	3596	**3343**	-7,04%
	8	4048	**3651**	-9,81%
	9	4107	**3982**	-3,04%
	10	3963	**3809**	-3,89%
				-5,44%

5 STATISTICAL ANALYSIS

5.1 Statistical analysis of the randomly-generated problems

In order to discover the relationship between the two algorithms and to obtain more reliable test results, the data was transferred to SPSS software. First, descriptive statistics were gathered for μ_D (mean differences) for six test sets. The results are given in Table 8 where, as can be seen, negative differences show that LST-LA reaches lower maximum lateness values than EDD.

The paired t test is performed for all groups to see whether there is a significant difference between the two algorithms [18].

H_o: $\mu_D=0$
H_1: $\mu_D\neq0$

As can be seen from Table 9, p value = 0.00 <0.05 and H_0 is rejected at level 0.05.

Table 8: Descriptive statistics for six groups

Job size	N	Mean	Std. deviation	Std. error	95% confidence interval for mean		Minimum	Maximum
					Lower bound	Upper bound		
10	15	-32.87	25.108	6.483	-46.77	-18.96	-72	17
30	15	-61.33	54.395	14.045	-91.46	-31.21	-188	24
50	10	-62.70	78.284	24.756	-118.70	-6.70	-186	40
80	10	-183.40	78.774	24.911	-239.75	-127.05	-296	-71
100	10	-181.40	118.756	37.554	-266.35	-96.45	-371	-22
150	10	-218.40	154.802	48.953	-329.14	-107.66	-455	-56
Total	70	-112.46	112.478	13.444	-139.28	-85.64	-455	40

Table 9: Paired samples test for problem size 10

		Paired differences					t	Df	P value
		Mean	Std. dev.	Std. error mean	95% confidence interval of the difference				
					Lower	Upper			
Pair 1	L_{max}EDD - L_{max}LTSLA	32.867	25.108	6.483	18.962	46.771	5.07	14	.000

Tukey's procedures are then used to look for significant differences among μ_{D_i}s [17]. The ANNOVA table (Table 10) is also used to see whether there is a significant difference between the results of LST-LA and EDD, both between and within groups.

Hypothesis testing is:

H_o: All μ_{D_i}s are equal

H_1: At least two of the μ_{D_i}s are different

Table 10: ANOVA Table

	Sum of squares	df	Mean square	F	P value
Between groups	369081.005	5	73816.201	9.376	.000
Within groups	503852.367	64	7872.693		
Total	872933.371	69			

Tukey's test results can be seen in Table 11.

Table 11: Tukey difference

Size	N	Subset for alpha = 0.05	
		1	2
150	10	-218.40	
80	10	-183.40	
100	10	-181.40	
50	10		-62.70
30	15		-61.33
10	15		-32.87
Sig.		.920	.967

Group sizes 10, 30, and 50 are not significantly different from each other. Group sizes 80,100, and 150 are also not significantly different from one another, but these two groups are significantly different from each other, as seen in Table 11. This result shows that when the number of jobs increases, LST-LA outperforms EDD on minimising the maximum lateness problem.

5.2 Experimental tests on Carlier's benchmark problems

For the additional experimental analysis, test instances were generated in the way defined by Carlier [16]. For the generation of problems, the following parameters were used:

n= 50, 100,150,200, …, 1000, and
K= 16, 17, 18, 19, 20, 21, 22, 23, 24, 25
A sample of 20 instances was generated for each job size and difficulty level.
p_j is generated from a discrete uniform distribution between 0 and p_{max}= 50,
r_j is generated from a discrete uniform distribution between 0 and r_{max} =n*K
q_j is generated from a discrete uniform distribution between 0 and q_{max}=n*K
$d_j = r_j+p_j+q_j$

where n is the job size, p_j is the processing time of the job j, r_j is the release date of job j, K is the parameter to define the tightness of due dates, q_j is the tail time of job j, p_{max} is the maximum processing time of jobs, r_{max} is the maximum release time, q_{max} is the maximum tail time, and d_j is the due date of job j.

In this case, 4,000 instances were analysed in 200 different problem sets; the average maximum lateness values obtained by LST-LA and EDD are summarised in Table 12. A straightforward comparison between LST-LA and EDD for Carlier's benchmark problem shows that LST-LA outperforms EDD. For example, for the job size of 50 and difficulty level of 16, the average difference between LST-LA and EDD rules is 480.18 per cent.

6 CONCLUSION

Considering the NP-hardness of the $1|rj|L_{max}$ scheduling problem, the heuristic algorithm called LST-LA was developed to obtain a better solution than the EDD algorithm. The algorithm was first tested on six different job size groups, which were 10, 30, 50, 80,100, and 150 jobs. These job size groups were generated randomly with defined parameters, as presented in Section 4, and the results were tested in SPSS. For each set, computational results show that LST-LA outperforms EDD and that the highest improvement over the EDD rule is obtained on the problems with 10 jobs.

In order to see the performance of the proposed LST-LA algorithm, it was also tested on randomly-generated Carlier's instances, as explained in Section 5.2. In these instances, LST-LA outperformed the EDD rule, even on the hard problems with difficulty levels of K= 18, 19, and 20.

The main contribution of the LST-LA algorithm is that it solves the $1|r_j|L_{max}$ problem as easily as the EDD rule, but with an improved solution performance. This proposed algorithm can be used easily in practice in a make-to-order environment that uses a single machine (e.g., plastic injection machine, moulding machine, or press machine). The proposed algorithm (LST-LA) can also be used to solve more complex scheduling problems, such as the shifting bottleneck algorithm, which iteratively solves a job shop scheduling problem by considering each machine as a single machine sub-problem.

For future work, we plan to use the LST-LA algorithm to boost the performance of the algorithms that solve a more complicated scheduling problem, by partitioning it to single machine sub-problems.

Table 12: Average maximum lateness values of LST-LA and EDD for 4,000 problem sets of Carlier's instances

Job size		K value									
		16	17	18	19	20	21	22	23	24	25
50	LST-LA	75,8	11,8	17,3	8,2	1,8	-5,2	-1,4	-3,5	1,3	-12,5
	EDD	251,7	153,6	163,2	172,6	109,4	89,4	100,5	60,1	65,5	32
100	LST-LA	54,6	9,8	-0,8	-6,5	-1	-4,6	0,5	-9	-5,2	-10,5
	EDD	463,9	346,2	270,6	270,1	171,6	146,6	134,8	87,2	87	30,6
150	LST-LA	60,9	0,1	3,8	3	-3,1	6,9	-5,4	-7,3	-7,4	2
	EDD	647,8	462,2	462,7	277,5	249,6	193	178,1	121	74,7	111,5
200	LST-LA	19,6	4,2	2,2	5,9	2,8	-0,1	-7,9	3,3	-5	-6,4
	EDD	812,3	671,3	504,2	429,5	329,7	276,7	148,3	133,8	69,9	54,4
250	LST-LA	19,2	0,1	-1,8	-3	2,9	-1	-2,8	-0,3	-1,8	-7,4
	EDD	981,1	860,6	677,3	471,1	390,6	303,6	200,9	231,5	165,6	113,8
300	LST-LA	35,4	4,4	3,4	7,8	-4,1	1,7	4,9	-0,7	-14,1	-3,1
	EDD	1185,7	986,9	787,9	675	385	373,2	277,7	208,3	78,2	129,2
350	LST-LA	14,9	3,9	7,3	8,4	1,9	9,2	-8,5	-9,5	1,5	-9,9
	EDD	1260,5	1029,8	966,8	690,8	688,6	510,7	255,5	246,3	147,1	86,4
400	LST-LA	15,2	10	3,8	2,7	-15,5	-4,9	0,1	-0,6	-8	-2,1
	EDD	1658,7	1188	1026,7	720,5	569,4	512,7	370,3	205,9	144,8	154,2
450	LST-LA	9,6	3,4	-0,9	1,1	-5,1	0,6	-6,6	-1	-14,2	2,2
	EDD	1617,3	1513,9	1001,8	914,3	646,9	488,8	385,6	292,9	139,1	187,5
500	LST-LA	18,1	4,9	3,1	1,7	8,4	-9,8	-3,3	-4,3	-12,1	-8,5
	EDD	2033,4	1608,5	1229,3	1034,4	701,6	435	460,9	302,7	153,9	163,7
550	LST-LA	1,8	6,6	5,7	6,9	-3,9	0,9	-1,8	-1,2	-8,6	-6,3
	EDD	1959,3	1743,2	1232,5	1031,4	896,5	624,1	389,1	295,4	173,6	133,4
600	LST-LA	9,7	4,7	5,6	10,2	9,9	-0,2	-0,8	-1,8	0,9	-17,9
	EDD	2368,8	1815,9	1560,3	1221,6	929,5	646,5	515	280,5	253	165
650	LST-LA	6,1	6,1	0,6	-6,4	4,6	2,9	-2,9	1,7	0,3	-16,9
	EDD	2403	2010,1	1640,8	1250	854,5	757,8	450,7	348,9	250,7	88,3
700	LST-LA	15,4	-1,4	4,5	-0,3	-10,6	-3,4	2,5	-6,6	-3,3	0,9
	EDD	2652,4	2244,3	1658,7	1357,3	970,3	815,4	532,2	286,7	218,8	165
750	LST-LA	21,4	-4,8	6,6	4,6	-9,1	2,1	-7,3	-2,4	-9,9	-17,1
	EDD	2921,3	2341,1	1816,5	1428,8	1162,6	766	600,3	384	164,3	221,1
800	LST-LA	9,3	0,8	2,7	1,8	-4,9	2,9	8	-2,8	-21,2	-2,6
	EDD	3109	2256,1	1811,4	1414,5	1159,7	857,2	640,3	436,4	274,5	195,6
850	LST-LA	10,8	2,6	3,9	3,8	1,2	-4,2	-7,1	-3,3	-5,9	-5,8
	EDD	3177,1	2467,8	2227,4	1562,4	1201	988,4	540,7	538,4	191	229,3
900	LST-LA	13,9	-3,8	-6	4	-5,5	2,3	-5,3	-3,8	-6,8	-1,1
	EDD	3458,9	2718,9	2158,5	1821	1227,1	983,1	685,3	440,8	302,2	187,4
950	LST-LA	4,9	-2,8	6,6	7,9	8	1,6	-2,5	-9,8	-12,7	-0,8
	EDD	3542,4	2737,5	2289,3	1645,8	1476,3	858,4	730,8	447,1	269,1	196
1000	LST-LA	3,4	2	3,1	4,7	2,5	7,3	-8,6	-6	-13,9	-2,1
	EDD	3841,6	3120,8	2487,3	1728,5	1488,3	886,9	606,3	412	312,1	189,2

REFERENCES

[1] **Chang, P.C. and Su, L.H.** 2001. Scheduling n jobs on one machine to minimize the maximum lateness with a minimum number of tardy jobs. *Computer and Industrial Engineering*, 40(4), pp. 349-360.

[2] **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.

[3] **Lenstra, J.K., Rinnooy, Kan, A.H.G. & Brucker, P.** 1977. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1, pp. 343–362.

[4] **Liu, L. and Zhou, H.** 2012. Applying variable neighborhood search to the single-machine maximum lateness rescheduling problem, *Electronic Notes in Discrete Mathematics*, 39, pp. 107-114.

[5] **Baker, K.R. and Magazine, M.J.** 2000. Minimizing maximum lateness with job families. *European Journal of Operational Research*, 127(1), pp. 126-139.

[6] **Sels, V. and Vanhoucke, M.** 2011. A hybrid dual-population genetic algorithm for the single machine maximum lateness problem. *Lecture Notes in Computer Science*, *6*622, pp. 14-25.

[7] **Oyetunji, E.O. and Oluleye, A.E.** 2008. Heuristics for minimizing the number of tardy jobs on a single machine with release time. *South African Journal of Industrial Engineering*, 19(2), pp.183-196.

[8] **McMahon, G. and Florian, M.** 1975. On scheduling with ready times and due dates to minimize maximum lateness. *Operations Research*, 23(3), pp. 475-482.

[9] **Lageweg, B.J., Lenstra, J.K., and Rinnooy-Kan, A.H.G.** 1976. Minimizing maximum lateness on one machine: Computational experience and some applications. *Statistica Neerlandica,* 30(1), pp. 25-41.
[10] **Frederickson, G.N.** 1983. Scheduling unit-time tasks with integer release times and deadlines. *Information Processing Letters*, 16(4), pp. 171–173.
[11] **Baker, K.R., Lawler, E.L., Lenstra, J.K., and Rinnooy-Kan, A.H.G.** 1983. Preemptive scheduling of a single machine to minimize maximum cost subject to release dates and precedence constraints. *Operations Research,* 31(2), pp. 381-386.
[12] **Gordon, V.S.** 1993. A note on optimal assignment of slack due-dates in single-machine scheduling. *European Journal of Operational Research,* 70(3), pp. 311–315.
[13] **Oyetunji, E.O. and Oluleye, A.E.** 2010. New heuristics for minimising total completion time and the number of tardy jobs criteria on a single machine with release time. *South African Journal of Industrial Engineering*, 21(2), pp. 101-113
[14] **Monma, C. and Potts, C.** 1989. On the complexity of scheduling with batch setup times. *Operations Research,* 37(5), pp. 798–804.
[15] **Schrage, L.E.** 1971. *Obtaining optimal solution to resource constrained network scheduling problems*. Unpublished manuscript, 189.
[16] **Carlier, J.** 1982. The one-machine sequencing problem, *European Journal of Operational Research,* 11(1), pp.42-47.
[17] **Pinedo, M.** 2002. *Scheduling theory algorithms and systems*. 4th edition, Prentice Hall.
[18] **Jackson, J.R.** 1955. *Scheduling a production line to minimize maximum tardiness*. Research Report 43, Management Science Research Project, University of California, Los Angeles, CA.
[19] **Potts, C.N. and Strusevich, V.A.** 2009. Fifty years of scheduling: A survey of milestones. *The Journal of the Operational Research Society,* 60(5)(Supplement), pp. 41-68.
[20] **Devore, J.L.** 1995. *Probability and statistics for engineering and sciences*. 4th edition, Wadsword Inc.