Int J Adv Manuf Technol (2003) 22: 243–248
DOI 10.1007/s00170-002-1466-0

ORIGINAL ARTICLE

R. Maheswaran · S. G. Ponnambalam

An investigation on single machine total weighted tardiness scheduling problems

Received: 3 April 2002 / Accepted: 26 June 2002 / Published online: 11 June 2003

Abstract The urge to produce products in the production line at a faster rate with no compromise in quality has led to scheduling gaining greater importance in the modern day industries. Scheduling is concerned with the allocation of limited resources to tasks over time. The investigations on various scheduling problems have been of constant interest to researchers worldwide. This paper deals with an investigation on the total weighted tardiness of the single machine scheduling problems. The problems are solved by a heuristic procedure, namely backward forward heuristics. The different methods of formulating the problem instances are discussed. The benchmark problems and their best known values available in the OR library are used for the comparison to find out the influence of Relative Due Date (RDD) and Tardiness Factor (TF) used to generate the problem instances.

Notations used

n	Number of jobs
p_i	Processing Time of i^{th} job
d_i	Due date of i^{th} job
w_i	Weight of i^{th} job
p_m	Mean Processing Time
d_m	Mean Due date
σ	Complete sequence of jobs
$p_{[i]}$	Processing Time of the job in the i^{th} position in the sequence
$d_{[i]}$	Due date of the job in the i^{th} position in the sequence
$w_{[i]}$	Weight of the job in the i^{th} position in the sequence
$T_{[i]}$	Tardiness of the job in the i^{th} position in the sequence
$Z(\sigma)$	Total Weighted Tardiness of the sequence
p^l	Lower limit of processing Time
p^u	Upper limit of processing Time
d^l	Lower limit of due date
d^u	Upper limit of due date
RDD	Relative Due Date
TF	Tardiness factor index
T	Sum of the processing time of the jobs yet to be scheduled

R. Maheswaran
Department of Mechanical Engineering,
MEPCO Schlenk Engineering College,
Mepco Nagar-626 005,
Virudhunagar district, Tamil Nadu, India

S. G. Ponnambalam (✉)
School of Engineering and Science,
Monash University Malaysia,
46780, Kelana Jaya, Selangor, Malaysia
E-mail: sgponnambalam@yahoo.com

1 Introduction

Scheduling is concerned with the allocation of limited resources to tasks over time. It is a decision-making process that has as a goal to optimise one or more objectives. The resources and tasks may take many forms. The resources may be machines in a workshop, runways at an airport and crews at a construction site, as well as processing units in a computing environment, and so on. The tasks may be operations in a production process, take-offs and landings at an airport, stages in a construction project, executions of computer programs, and so on. Each task may have a different priority level, processing time, and due date. The objectives of scheduling may also take many forms. Some possible objectives are the minimisation of the completion time of the last task, the minimisation of the number of tasks completed after the committed due dates etc. If the job is completed after its due date, it is considered as Tardy (Baker 1974), and charged with a penalty equal to its completed time minus due date known as weighed tardiness. The sum of the weighted tardiness of all jobs in a sequence is known as total weighted tardiness. The problem environment we have

chosen is a Single Machine-Scheduling with performance measure of total weighted tardiness. The objective is to find a sequence that minimises the total weighted tardiness.

Bahram Alidaee and Ramakrishnan states the same problem as follows (1996): There is a set of n independent jobs ready at time zero to be scheduled on a single machine which is continuously available. Associated with a job i ($i=1,2,.....,n$) there is a processing time (p_i) and due date (d_i) and a weight (w_i). Let $\sigma=([1], [2],.....,[n])$ be a sequence of the jobs, where $[i]$ is the i^{th} job. Given a sequence σ, let

$$C_{[i]} = \sum_{j=1}^{i} p_{[j]} \tag{1}$$

$$T_{[i]} = \max\left\{0, C_{[i]} - d_{[i]}\right\} \tag{2}$$

be the completion time and tardiness of i^{th} job. The objective is to find a sequence σ that minimises

$$Z(\sigma) = \sum_{i=1}^{n} w_{[i]} T_{[i]} \tag{3}$$

Lenstra et al. proved that single machine total weighted tardiness problems are strongly NP hard (Lawler et al. 1993). Coffman observed that complete enumeration to get exact solution, have computational requirements that frequently grow as an exponential or high-degree polynomial in the problem size n. In such cases, problems of practical size cannot be solved exactly and some approximate solution with guaranteed accuracy can be obtained by certain heuristic algorithms (1976). One such heuristic algorithm is used for our investigation to find out the influence of the hardness determining factors used to generate the bench mark problem instances available in the OR library.

2 Test problem generation

There is no common method for the formulation of problem instances for the single machine scheduling. Many authors used different random methods for generating the problems as explained below.

Hariri et al. generated the test problem instances as follows (1995): They used to select two parameters namely, lower bound of due date (d^l) and upper bound of due date (d^u) which represent the relative values of due dates. For each job, an integer value of processing time (p_i) is randomly generated from the uniform distribution [1,100] and an integer weight (w_i) is randomly generated from the uniform distribution [1,10]. After selecting the parameters d^l and d^u, total processing time $P = \sum_{i=1}^{n} p_i$ is computed and an integer due date (d_i) is generated from the uniform distribution $[P\ x\ d^l, P\ x\ d^u]$.

Crauwels et al. generated the test problem instances as follows (1998): They used to select two parameters namely relative range of due date (RDD) and the average tardiness factor (TF). For each job, the integer processing time (p_i) is generated from the uniform distribution [1,100] and an integer weight (w_i) is generated from uniform distribution [1,10]. For the selected values of RDD and TF total processing time $P = \sum_{i=1}^{n} p_i$ is computed and an integer due date (d_i) is generated from the uniform distribution $[P(1-TF-RDD/2),\ P(1-TF+RDD/2)]$.

Franca, Mendes and Moscato generated the test problem instances (2001). The generation of processing times and set up times of each job follows a discrete uniform distribution DU[0,100] according to two parameter namely due date range (Δd) and due date mean (d_m). The due date range is defined by due date factor (R) which is described as $\Delta d = R \times n \times p_m$ and due date mean is defined by Tardiness Factor (TF) which is described by $d_m = (1-TF) \times n \times p_m$.

3 The heuristic algorithm used

Dileep Sule described a heuristic procedure which is easy to apply and is efficient called as Backward Forward heuristics. This procedure is developed in two phases: the backward phase is applied first, to get an initial sequence and then the forward phase is applied to make further improvements (1997).

3.1 The backward phase

In the backward phase the initial sequence is developed by the following procedure. The sequential job assignment starts from the last position and proceed backward towards the first position. The assignments are complete when the first position is assigned a job. The process consists of the following steps:

Step 1: Note the position in the sequence in which the next job is to be assigned. The sequence is developed starting from position n (number of jobs) and continuing backward to position 1. So, the initial value of the position counter is n.

Step 2: Calculate T, which is the sum of the processing times for all unscheduled jobs.

Step 3: Calculate the penalty for each unscheduled job i as $(T-d_i)\times w_i$. If $d_i > T$, the penalty is zero, because only tardiness penalties are considered.

Step 4: The next job to be scheduled in the designated position is the one having the minimum penalty from step 3. In the case of tie, choose the job with the largest processing time.

Step 5: Reduce the position counter by1.

Repeat steps 1 through 5 until all jobs are scheduled.

3.2 The forward phase:

Perform the forward phase on the job sequence found in the backward phase that is the best sequence at this stage. The forward phase progresses from the job position 1 towards the job position n. Let k define the lag between two jobs in the sequence that are exchanged. For example, jobs occupying positions 1 and 3 have a lag $k=2$. The forward phase algorithm is described by means of a flowchart as shown in Fig. 1.

4 The computational experience

The bench mark problems are available in OR library from the web site http://www.ms.ic.ac.uk/jeb/orlib/wtinfo.html and the problem instances are generated as follows:

For each job j ($j=1,...,n$), an integer processing time p_j was generated from the uniform distribution [1,100] and integer processing weight w_j was generated from the

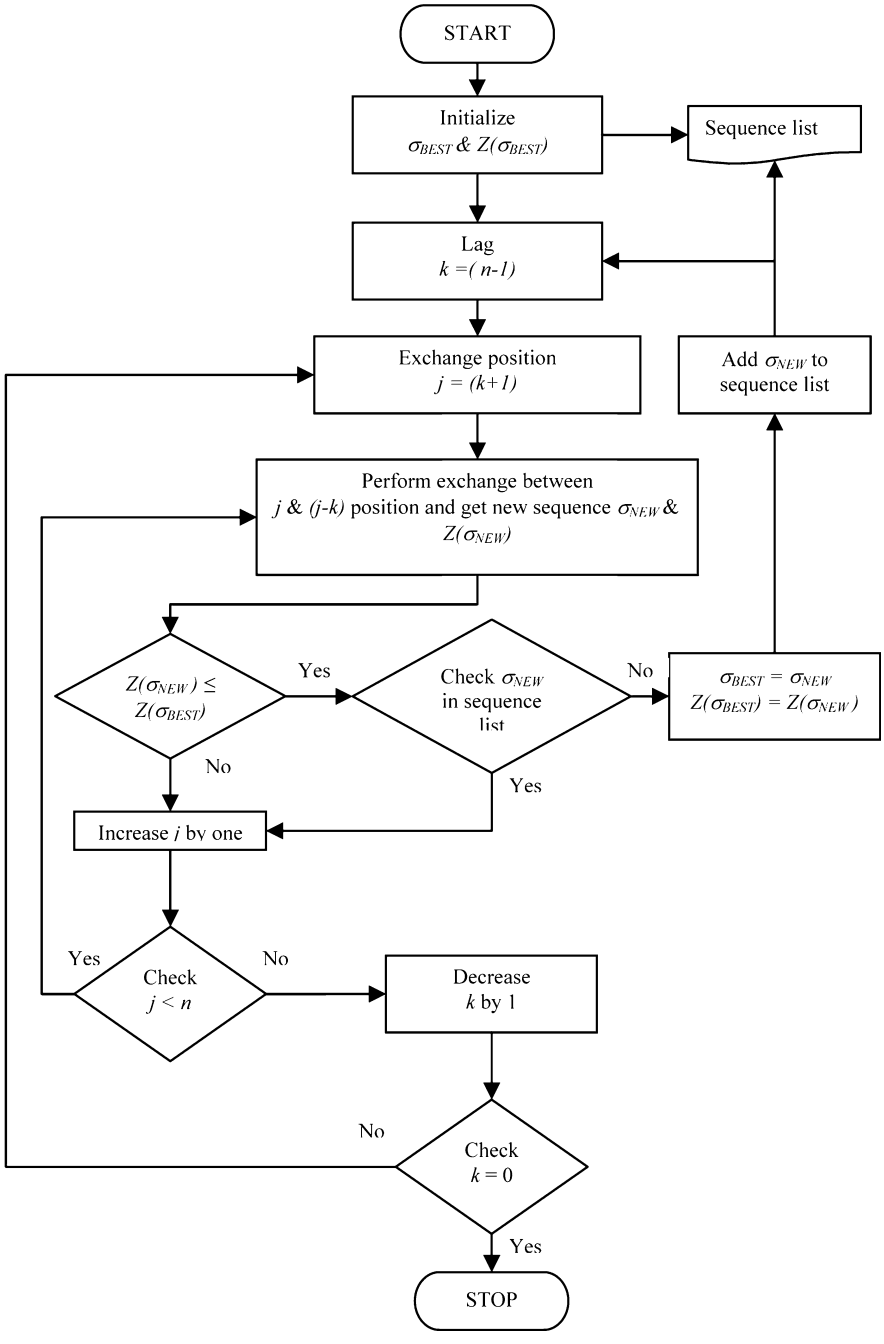

Fig. 1 Flow chart for the forward phase

Table 1 Comparison of the value with the best known value for $n=40$

S. No	RDD	TF	Best known value (OR-library)	Calculated value (BF heuristics)
1.	0.2	0.2	913	913
2.		0.4	16225	16668
3.		0.6	17465	17562
4.		0.8	63229	64272
5.		1.0	119947	120536
6.	0.4	0.2	64	64
7.		0.4	6575	6709
8.		0.6	19611	20479
9.		0.8	78139	78830
10.		1.0	113999	114598
11.	0.6	0.2	0	0
12.		0.4	3784	3784
13.		0.6	19771	20280
14.		0.8	78451	78889
15.		1.0	115249	116691
16.	0.8	0.2	0	0
17.		0.4	684	684
18.		0.6	25881	26743
19.		0.8	55730	56490
20.		1.0	114686	115109
21.	1.0	0.2	0	0
22.		0.4	0	0
23.		0.6	21109	21380
24.		0.8	66707	67138
25.		1.0	73041	73252

Table 2 Comparison of the value with the best known value for $n=50$

S. No	RDD	TF	Best known values (OR-library)	Calculated value (BF heuristics)
1.	0.2	0.2	1996	2069
2.		0.4	8499	8716
3.		0.6	36378	36542
4.		0.8	72316	74145
5.		1.0	224025	224455
6.	0.4	0.2	4	4
7.		0.4	9934	10124
8.		0.6	22739	23041
9.		0.8	90163	91680
10.		1.0	133289	133497
11.	0.6	0.2	0	0
12.		0.4	3770	3926
13.		0.6	17337	17600
14.		0.8	77930	78745
15.		1.0	98494	99187
16.	0.8	0.2	0	0
17.		0.4	816	816
18.		0.6	15451	16707
19.		0.8	89289	90264
20.		1.0	139591	140298
21.	1.0	0.2	0	0
22.		0.4	0	0
23.		0.6	35106	36787
24.		0.8	101665	102319
25.		1.0	78315	78852

uniform distribution [1,10]. Instance classes of varying hardness were generated by using different uniform distributions for generating the due dates. For a given relative range of due dates RDD (RDD = 0.2, 0.4, 0.6, 0.8, 1.0) and a given average tardiness factor TF ($TF=0.2$, 0.4, 0.6, 0.8, 1.0), an integer due date d_j for each job j was randomly generated from the uniform distribution $[P(1-\mathrm{TF}-RDD/2),\ P(1-TF+RDD/2)]$, where $P=SUM\{j=1,...,n\}\times p_j$.

For the Backward Forward heuristics, a C++ code is developed and executed in the Pentium II 233 MHz processor, 128 MB ram. We tested the program on 125 benchmark instances for the single machine total weighted tardiness problems of number of jobs $n=40$,

Table 3 Comparison of the value with the best known value for $n = 100$

S No	RDD	TF	Best known values (OR-library)	Calculated value (BF heuristics)
1.	0.2	0.2	5283	5470
2.		0.4	59434	61579
3.		0.6	157476	166619
4.		0.8	544838	577793
5.		1.0	744287	764853
6.	0.4	0.2	8	8
7.		0.4	24202	25299
8.		0.6	90440	98936
9.		0.8	425875	445268
10.		1.0	623356	649886
11.	0.6	0.2	0	0
12.		0.4	19912	21412
13.		0.6	56510	58655
14.		0.8	401023	437133
15.		1.0	640816	658963
16.	0.8	0.2	0	0
17.		0.4	1400	1400
18.		0.6	54612	58877
19.		0.8	326258	340573
20.		1.0	622464	652952
21.	1.0	0.2	0	0
22.		0.4	0	0
23.		0.6	159123	167277
24.		0.8	370614	397859
25.		1.0	397029	420450

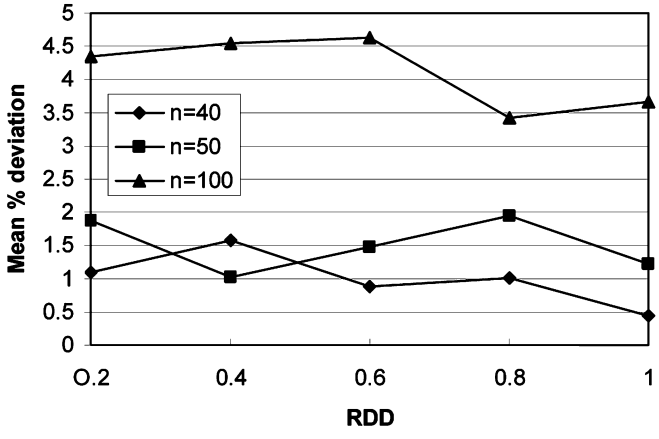

Fig. 2 Mean deviation vs. RDD

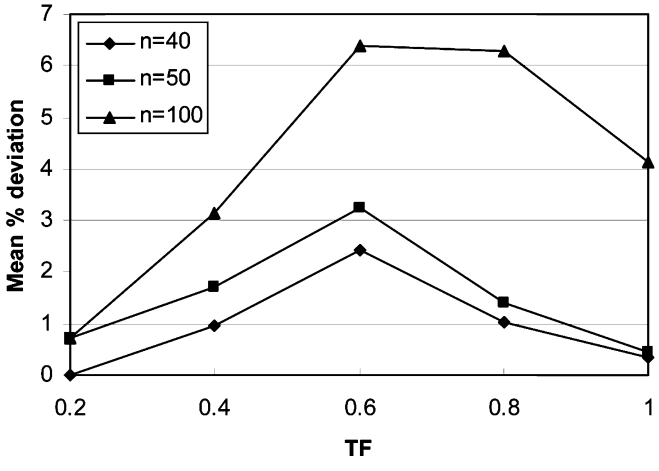

Fig. 3 Mean percentage deviation vs. TF

Table 4 Maximum and minimum mean deviations

Number of jobs	Minimum mean deviation		Maximum mean deviation	
	RDD	TF	RDD	TF
n = 40	0.6	0.2	0.4	0.6
n = 50	0.4	0.2	0.8	0.6
n = 100	0.8	0.2	0.6	0.6

$n = 50$ & $n = 100$. The bench mark set contains five instances of each combination of *TF* & *RDD* values. After having pilot runs, one best result from each set which leads to 25 problems in each case of $n = 40$, $n = 50$ & $n = 100$ are analysed and the results are compared with the available best known values. The comparison is shown in the Tables 1, 2 and 3.

All of each instance of the bench mark instances are solved in terms of milliseconds. The percentage of deviation for the algorithm from the best known values are calculated by the following formula,

$$\% \text{ of deviation} = \frac{Z_{cal} - Z_{best}}{Z_{best}} \times 100$$

Where Z_{cal}–Calculated weighted tardiness value by algorithms Z_{best}–Best known weighted tardiness value available in OR library.

The mean percentage of deviation for the test problems are found to be 0.998% for $n = 40$, 1.51% for $n = 50$ and 4.114% for $n = 100$ jobs.

The average percentage of deviation for the different *RDD* values of the bench mark problems with the numbers of jobs $n = 40$, 50 & 100 is calculated and compared and shown in Fig. 2

The average percentage of deviation for the different *TF* values are calculated and compared and shown in Fig. 3

From the above graphs, the maximum and minimum mean deviations occurring for the different values of the *RDD* and *TF* are shown in Table 4.

From Table 4 of the minimum mean deviation of the obtained results by backward forward heuristics from the best known solution, it is observed that for all the set of bench mark problems is occurring at the *TF* value of (0.2) and the maximum mean deviation is occurring at the *TF* value of (0.6).

5 Conclusions

This paper deals with the investigations of the performance of Backward Forward heuristic algorithm for solving the problem of scheduling in a single machine to minimise the total weighted tardiness. Three different methods of generation of test problem instances are discussed. Extensive computational procedure is used to compare the algorithm by solving the problem instances available in the OR library and the best known value is used as the criteria for finding out the percentage of deviation. As the CPU time for getting the best known values are not available and the time taken to solve even 100 jobs problem is within the acceptable limit of millisecond, the comparison of CPU time is not given in this paper. The solution obtained by the proposed heuristics is within minimum percentage of deviation from the optimal or best known solution, and it may be considered as a guaranteed approximate solution.

Acknowledgement The authors are grateful to the management, the principal and the head of the Mechanical Engineering Department of Mepco Schlenk Engineering College for their kind support of this paper.

References

Alidaee B, Ramakrishnan KR (1996) A computational experiment of covert-AU class of rules for single machine tardiness scheduling problems. Comp Ind Engin J 30(2):201–209

Baker KR (1974) An introduction to sequencing and scheduling. Wiley, New York

Coffman JR (1976) Computer and job-shop scheduling theory. Wiley, New York

Crauwels HAJ, Potts CN, van Wassenhove LN (1998) Local search heuristics for the single machine total weighted tardiness scheduling problem. J Comput 10(3):341–350

Franca PM, Mendes A, Moscato P (2001) A memetic algorithm for the total tardiness single machine scheduling problem. Eur J Operat Res 132(1):224–242

Hariri AMA, Potts CN, van Wassenhove LN (1995) Single machine scheduling to minimize total weighted late work. J Comput 7(2):232–242

Lawler EL, Lenstra JK, Alexander HG, Kan R, Shmoys D (1993) Sequencing and scheduling: algorithms and complexity. Handbooks in OR & M.S, vol. 4, Elsevier, Amsterdam

Sule DR (1997) Industrial scheduling. PWS, Boston, MA