Myopic Heuristics for the Weighted Tardiness Problem on Identical Parallel Machines

Ram Mohan V. Rachamadugu and Thomas E. Morton

CMU-RI-TR-83-17

University of Michigan
Ann Arbor, MI

Carnegie-Mellon University
Pittsburgh, PA

15 August 1983

Abstract

We study the problem of scheduling jobs on parallel identical machines to minimize weighted tardiness. There are no known heuristics for this problem. The heuristic rule developed by us is simple. It can be used in the dispatch mode which makes it very practical. Characterizations of optimal solutions are presented. Computational results show that the myopic heuristic performed well vis-a-vis other rules such as the Earliest Due Date Rule, Weighted Shortest Processing Time Rule and two versions of Montagne's Rule. We report computational results and also the performance of the heuristic in comparison to high computational benchmarks.

MYOPIC HEURISTICS FOR THE WEIGHTED TARDINESS PROBLEM ON IDENTICAL PARALLEL MACHINES

1. Introduction

The problem of minimizing weighted tardiness of a given set of jobs, available simultaneously and to be scheduled without pre-emption, has attracted the attention of several researchers. It has been shown by Lenstra that in the case of single machine this problem is NP-complete. In view of the importance of this problem, earlier studies on this problem emphasized on the development of effective enumerative and heuristic procedures. In a recent study Morton and Rachamadugu [14] developed a myopic heuristic for this problem which can be used in the dispatch mode. Our heuristic extends the myopic heuristic developed by Morton and Rachamadugu to the weighted tardiness problem in the case of identical parallel machines. There are no known effective heuristics for this problem. Briefly, the problem is as follows: we have n jobs $J_1, J_2, J_3 \ldots\ldots J_n$ that arrive simultaneously to be processed on a set of identical parallel machines. The jobs are not permitted to be pre-empted and no job can be processed on more than one machine. Each job is characterized by the triple(p_i, d_i, w_i) which represents the processing time, due date and the weight(tardiness penalty) of the job. Each job has associated with it the penalty function $C_i(t_i)$ where t_i is the completion time of the job. $C_i(t_i)$ is given by

$$C_i(t_i) = w_i \max\{0,\ t_i - d_i\}$$

We wish to allocate the jobs among the machines and schedule them such that $\Sigma C_i(t_i)$ is minimized.

2. Review of prior studies

As pointed out by Graves [8] no simple results exist for the weighted tardiness problem in the case of parallel machines. Lawler [10] studied the problem (identical machines) for jobs with equal processing times and monotonically nondecreasing penalty cost of completion time and showed that the problem can be solved using the

transportation algorithm. For problems where jobs have unequal processing times, he suggested a procedure which yields a lower bound for the problem. Root [16] studied the case where job weights are equal and the jobs have the same due date. Other special cases were studied by Elmaghraby and Park [7], Nunnikhoven and Emmons [13] and Barnes and Brennan [2]. However, there do not seem to exist any heuristic procedures for the general case of the weighted tardiness problem for parallel machines. Though one can attempt a dynamic programming approach for finding the optimal solution, such a procedure is very severely limited by 'the curse of dimensionality'.

Rajaraman [15] developed an enumerative algorithm for optimal scheduling of jobs on identical parallel processors when the jobs have nondecreasing waiting costs and also extended it to the case when jobs have due dates. However, Dogramaci and Sukris [5] have shown that the procedure is in error and provided additional sufficiency conditions to make the algorithm valid. However, they concede that the imposition of the sufficiency conditions leads to an excessive computational burden. Bernardo and Lin [3] developed a heuristic procedure for the average tardiness problems in the case of parallel processors. The limitation of their heuristic is that in the case of single machine problems, it reduces to the Earliest Due Date rule. Morton and Rachamadugu [14] have shown that the Earliest Due Date rule performs rather badly in the case of single machine problems, particularly when the tardiness of the job set is high.

In this paper we study the weighted tardiness problem in the case of identical processors and extend the concept of a myopic heuristic for this problem. In developing and validating the heuristics, we are more interested in the average behavior of the heuristic procedure rather than the worst case behavior though the latter is of some significance. As pointed out by De and Morton [4], practitioners are typically interested in the 'average behavior' of the heuristic procedures. Thus, in testing the heuristics, we are interested in estimating the average deviation of the heuristic from

the optimum solution, if possible, or a 'high computational benchmark' such as truncated branch and bound or an effective lower bound. We derived effective lower bounds for 320 problems and compared the performance of the myopic heuristic against the lower bound. We also compared the myopic heuristic against competing heuristics in a large study consisting of 1280 problems. In our study, the myopic heuristic performed better than the competing heuristics in an average sense and provided solutions close to the optimum when used in a dispatch mode.

3 Myopic Heuristics for Parallel Processors

Consider the case in which the processors are identical. It is clear that in this case, no machine is idle in an optimal solution (n > m. If n ≤ m, the problem is trivially solved by assigning not more than one job to any machine.) Consider the relaxation of the problem in which the jobs have unit processing times. Let J_i and J_j be any two jobs on any two machines in an optimal solution such that J_i completes earlier than J_j. Let C_i be the completion time of J_i and C_j equal C_i+X. It can be easily seen that the following propositions, derived in [14] for the pre-emptive version of the single machine case, hold good for this case also:

PROPOSITION I : In the case of identical processors with pre-emption, the following property is satisfied by an optimal solution-

$$w_i \left[1 - \frac{(d_i - c_i)^+}{X} \right]^+ \geq w_j \left[1 - \frac{(d_j - c_i)^+}{X} \right]^+$$

PROOF :The proof is similar to the proof for the single machine case (Appendix [14]) and is omitted here for the sake of brevity.

The following proposition, valid in the case of any pair of adjacent jobs in the single machine problem, holds good for the optimal solution in the case of identical processors-

PROPOSITION II : For any pair of adjacent jobs on any processor, the following

condition must be satisfied by an optimal solution–

$$\frac{w_i}{p_i}\left[1 - \frac{\{d_i-(t+p_i)\}^+}{p_j}\right]^+ \geq \frac{w_j}{p_j}\left[1 - \frac{\{d_j-(t+p_j)\}^+}{p_i}\right]^+$$

where t is the start time for J_i and J_i immediately precedes J_j.

PROOF :The proof immediately follows from the proof in [14]. It can easily be seen that the Proposition II holds good also for the proportionate and unequal machine cases.

We note that the procedure for seeking optimal or good solutions to the weighted tardiness problem in the case of parallel machines involves two aspects: partitioning and ordering. In the case of the weighted completion time problems, for any given partition, best ordering can be found by using the weighted shortest processing time rule on each subset of jobs assigned to various processors. However, in the case of the weighted tardiness problems, optimal ordering by itself is a difficult task. In developing the myopic heuristic for this problem, it is clear that in an optimal solution, there is no inserted idle time. Also, while partitioning the jobs, we allocate the jobs to the machines such that the jobs with the highest 'urgency' are so assigned that they get completed as early as possible.

Proposition II can be used directly to find a schedule which cannot be improved by adjacent pairwise interchange on any machine. We exploit this propoerty in the following manner in developing our heuristic: at any instant a machine becomes available, we determine an apparent priority index(AP_i) for each of the unscheduled jobs as given below[1]:

$$AP_i = (w_i/p_i)\ \{\ 1\ -\ (d_i-t-p_i)^+/X\ \}^+$$

where t is the current time. Since we do not know which job is likely to

[1] we use the notation $X^+ = \max(0,X)$

suceed J_i in an optimal(or in any local minimum) solution, we approximate the processing time of the succeeding job by the mean processing time of all jobs or a multiple thereof. The apparent priority assigned to any job over time is shown in Figure 1. It is clear that if a job is too early, it need not be scheduled immediately. If the slack is zero or negative, then the job is assigned its full priority, w_i/p_i. The duration of intermidiate range is dependent on the value of X. It is also clear that as $X \to \infty$, our priority scheme reduces to the Weighted Shortest Processing Time Rule.

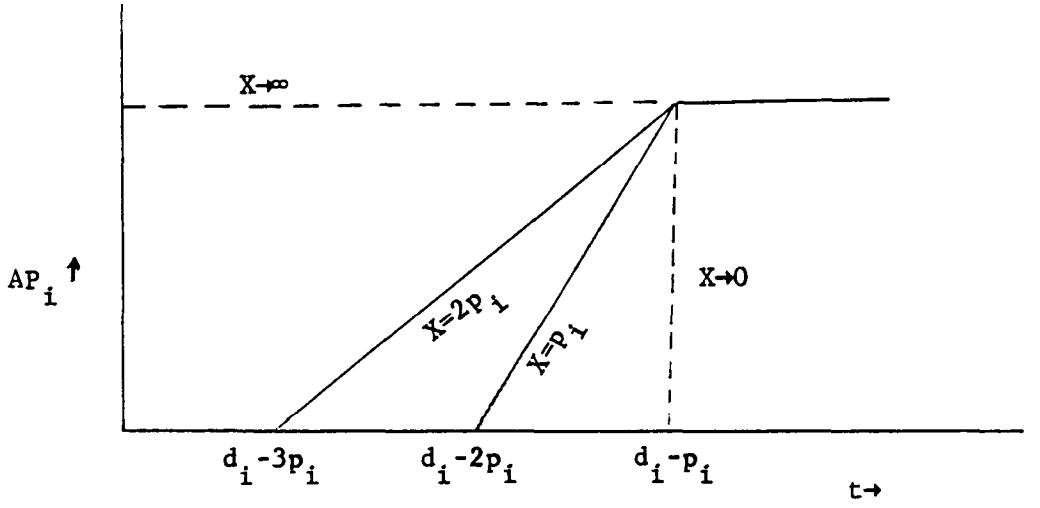


FIGURE 1

Taking into consideration the above observations, we devise the following myopic heuristics for the identical processors case–

MYOPIC HEURISTIC [H1]

Step 0 : Determine the apparent priority (AP_i) for all unscheduled jobs as follows:

$$AP_i = \frac{w_i}{p_i}\left[1 - \frac{\{d_i-(t+p_i)\}^+}{k\bar{P}}\right]^+$$

where t is the earliest time at which a processor becomes available(call it j), $\bar{P}$ is the mean processing time of the jobs and k is the parameter used in the heuristic. Go to Step 1.

Step 1 : Assign the job with the highest apparent priority to the first available machine and remove it from the list of unscheduled jobs. Go to Step 2

Step 2 : If the set of unscheduled jobs is empty, stop. Otherwise, go to Step 0.

For computational testing purposes, we used an exponential priority scheme. Under this procedure, apparent priority is given by

$$AP_i=(w_i/p_i)\ \exp\{(-k/\bar{p})(d_i-t-p_i)^+\ \}$$

Our earlier computational results in case of single machine problem showed that the exponent form performed better than the linear version [14].

It is to be noted that [H1] heuristic is a single-pass heuristic procedure. That is, we do not change the start time of a job once it is assigned. Secondly, partitioning and ordering decisions are simultaneously made. Thirdly, the rule is a dispatch procedure- i.e., *a procedure in which the actual decisions affecting a given machine can be implemented in the same order that they are made* [1]. There are many advantages to such procedures. They are generally simple and can easily be adapted to dynamic situations with minor modifications. The decisions are made in the same chronological order in which they are implemented. Further, in most practical situations, the manager is concerned with the immediate decision to be implemented. Dispatch procedures have the nice property that the computation can be terminated after the immediate decision is made and incorporate additional available information(such as new job arrivals and/or any change in the parameter values) in the decision framework

before the next decision to be implemented is made. As rightly pointed out by Morton and Dharan [12], managers may be willing to sacrifice the search for optimization for the ease and robustness of a dispatch rule. Dispatch rules, by and large, tend to be simple and are easily incorporated into more complex rules devised to take into account other operational constraints. Step 1 of the myopic heuristic ensures that the job with the highest apparent priority gets assigned immediately. At the first sight, it may appear that we can perhaps improve our results by reapplying the heuristic scheme for each subset of jobs assigned to various machines. Another way to improve the solution would be to use adjacent pairwise interchange, though this can at best ensure only local optimality of the schedule. However, any such improvement procedures destroy the dispatching nature of the heuristic and experimentally give only small improvements in our studies.

It can easily be seen that the Earliest Due Date rule and the Weighted Shortest Processing Time rule can be adapted for these problems. In the earliest due date rule, priority of a job is determined by the due date of the job– earlier the due date is, higher the priority. In the Weighted Shortest Processing Time rule, the priority is determined by the ratio w_i/p_i – higher the ratio is, higher the priority of the job. We also note that the Earliest Due Date rule and the Weighted shortest processing time rules are static rules– that is, the relative priority of the jobs do not change over time. Therefore, reapplication of the same heuristics for the jobs assigned to various machines does not alter the original schedule generated. However, the myopic heuristic is a dynamic rule in the sense that the relative priorities of the jobs may change over time.

Baker and Marin [1] in their paper on the experimental comparison of solution algorithms for the average(unweighted) tardiness problem refer to Montagne's method [11]. They claim it to be *very effective for the weighted version of the tardiness problem.* The heuristic is as follows: sequence the jobs in nondecreasing order of the ratio $p_i/(w_i*(\sum_{k=1}^{k=n} p_k - d_i))$– lower the ratio, higher the priority of the job.

This rule can also be used in the case of parallel machines and we call such a procedure 'Adapted Montagne's rule'. However, it is clear that adjustments can be made on the jobs assigned to any particular machine by reapplying the rule on the set of jobs assigned to that particular machine. We call this 'two-pass Montagne's Procedure'. In the two-pass procedure, we simply add the following step:

Step 4: Rearrange the jobs on each machine in nondecreasing order of the ratio $p_i/w_i*(\Sigma_{l \in N_k} p_l - d_i)$ where N_k is the set of jobs assigned to the machine k in Steps 0 through 3.

It may be noted that Adapted Montagne's rule is a static rule and can be used in dispatch mode. The two-pass Montagne's procedure is not a static rule and though it may yield better results than the simple version, it cannot be used as a dispatch procedure.

4. Design of the Computational Experiment

Control variables in the design of the computational experiment are the number of machines(m), number of jobs(expressed as the ratio of number of jobs per machine- n/m), the tardiness factor(τ) and the Range factor(R). The tardiness factor is an approximate measure of the number of jobs which may be expected to be tardy in a random sequence [17]. The range factor is a measure of the dispersion of the due dates of the jobs. For a detailed discussion of these two factors see [17].

- *weights for the jobs* : Since we expect that the penalty factor associated with the tardiness of a job to be correlated to the work content, it is expected that the weight of the jobs would be roughly proportionate to the processing time of the job. Taking this into consideration, we generate the ratio (w_i/p_i) for any job from the uniform distribution[0.,2.]

- *Processing Times* : Job processing times are generated from the Normal distribution with μ=30 and two values for the coefficient of variation-
  - Coefficient of variation for the processing times: 0.1,0.3

- *Due Dates* : In generating the due dates, two controlling factors are the tardiness factor(τ) and the range factor(R). As indicated earlier, the tardiness factor is a coarse measure of the proportion of jobs which might be expected to be tardy in a random sequence [17]. The range factor is again a coarse measure of the dispersion of the due dates from the mean due date of the given set of jobs. It is expressed as a ratio of the makespan of the problem. In the case of identical parallel processors case, we use the ratio $n\bar{p}/m$ as a measure of the makespan for generating the due dates. The values of the tardiness factor and the range factor used in the study are as given below:
  - Tardiness factor(τ): 0.2,0.4,0.6,0.8
  - Range factor(R): 0.4,0.8

  Job due dates are generated from a Normal distribution for the above values of tardiness factor and range factor.

- *Number of jobs per machine* : Number of jobs per machine are set at two levels- 15 jobs per machine and 30 jobs per machine.

- *Number of machines* : Number of machines were set at two levels- 2 and 5.

We tested 20 problems for each specification of the parameters. Thus, in total, we tested 2*4*2*2*2*20= 1280 problems. The heuristics tested were as follows:

1. Earliest Due Date Rule
2. Weighted Shortest Processing Time Rule
3. Adapted Montagne's procedure
4. Two-pass Montagne's procedure
5. Exponent form of the myopic heuristic

It may be noted that all the rules, except the 'Two-pass Montagne's procedure' are dispatch rules.

5. Computational Results

Since the study was conducted across a wide range of values of weights, processing times and the number of jobs, it is necessary to normalize the results for comparison and analysis. The metric that we use in our study is as follows:

$$\text{Performance measure of the rule} = \frac{\text{Weighted tardiness for the sequence generated by the rule}}{\bar{w}*n*\bar{p}}$$

$\bar{w}$,n and $\bar{p}$ are, respectively, the mean weight of the jobs, number of jobs and the mean processing time of the jobs in a problem. Division with the number of jobs makes results comparable across the problems with different number of jobs. Similarly, division with the mean weight normalizes the measure for differences in average weights. Finally, division with the average processing time expresses the measure in terms of **average number of average processing times tardy**.

The results of the computational experiment are shown in Tables I,II,III and IV. It is clear that the exponent form of the myopic heuristic performs better than the other heuristics. For comparison purposes, we tested the exponent form of the myopic heuristic with smoothing parameter value set at 1.0 against other heuristics.

Among the problems tested, the myopic heuristic performed better than any other heuristic in an average sense, except in the case of very low tardiness factor(τ=0.2) and high range value(R=0.8). Even in these cases, increasing the value of the smoothing parameter k led to a performance better than any other heuristic. Behavior of the myopic heuristic for varying values of the parameter k is illustrated for the case m=5, n/m=30 for varying values of the parameter in Figures 2 and 3. It is clear that the performance of the myopic heuristic can be improved by increasing

TABLE I

Mean values of Performance Measure for Heuristics

Number of machines: 2
Number of jobs per machine: 15

R	0.4					0.8				
τ	EDD	WSPT	M1	M2	H1	EDD	WSPT	M1	M2	H1
0.2	0.1135	0.0944	0.0874	0.0581	0.0382	0.0163	0.1320	0.1001	0.0230	0.0090
0.4	1.0085	0.4552	0.4364	0.4058	0.3286	0.6017	0.5637	0.4844	0.3511	0.1896
0.6	2.5070	1.2997	1.2724	1.2257	1.1121	2.1958	1.4814	1.3412	1.1669	0.9121
0.8	5.0060	3.0120	2.9728	2.9428	2.8488	5.0261	3.1517	3.0061	2.9084	2.7178

TABLE II

Number of machines: 2
Number of jobs per machine: 30

R	0.4					0.8				
τ	EDD	WSPT	M1	M2	H1	EDD	WSPT	M1	M2	H1
0.2	0.1195	0.1245	0.1142	0.0638	0.0306	0.0050	0.2231	0.1737	0.0296	0.0025
0.4	1.7513	0.7901	0.7572	0.6968	0.6432	0.9343	0.8766	0.7581	0.5231	0.2914
0.6	4.9485	2.4212	2.3061	2.2864	2.2345	4.1909	2.6885	2.4477	2.1832	1.7778
0.8	9.4954	5.4868	5.3947	5.3070	5.1162	9.6686	5.9662	5.6993	5.4797	5.0396

EDD Earliest Due Date Rule
WSPT Weighted Shortest Processing Time rule
M1 Adapted Montagne's method (dispatch version)
M2 Two-pass version of Montagne's method
H1 Exponent form of the myopic heuristic with parameter set at 1.0

TABLE III

Mean values of Performance Measure for Heuristics

Number of machines: 5
Number of jobs per machine: 15

R	0.4					0.8				
τ	EDD	WSPT	M1	M2	H1	EDD	WSPT	M1	M2	H1
0.2	0.1297	0.0865	0.0847	0.0565	0.0346	0.0032	0.1080	0.0999	0.0166	0.0051*
0.4	0.9839	0.4413	0.4365	0.3998	0.3089	0.6117	0.5332	0.5100	0.3500	0.1894
0.6	2.5311	1.3220	1.3100	1.2596	1.1152	2.2319	1.4712	1.4244	1.1920	0.8566
0.8	4.9591	2.9784	2.9561	2.9122	2.7647	4.9877	3.1411	3.0750	2.9233	2.6845

TABLE IV

Number of machines: 5
Number of jobs per machine: 30

R	0.4					0.8				
τ	EDD	WSPT	M1	M2	H1	EDD	WSPT	M1	M2	H1
0.2	0.1300	0.1179	0.1144	0.0681	0.0285	0.0001	0.2032	0.1874	0.0255	0.0011*
0.4	1.7140	0.7327	0.7224	0.6526	0.5507	0.9190	0.9130	0.8709	0.5308	0.2707
0.6	4.8413	2.3901	2.3714	2.2680	2.2111	3.9717	2.6173	2.5265	2.0972	1.6149
0.8	9.5990	5.5407	5.5031	5.3813	5.1435	9.3606	5.7673	5.6383	5.2774	4.7507

EDD Earliest Due Date Rule
WSPT Weighted Shortest Processing Time Rule
M1 Adapted Montagne's method (dispatch version)
M2 Two-pass version of Montagne's method
H1 Exponent form of the myopic heuristic with parameter set at 1.0
* Increasing the value of the parameter yields better results than the EDD rule

the value of the parameter value for low tardiness factor and/or high range values. It must be noted that the version of the myopic heuristic tested is a dispatch rule. In spite of this limitation, in our computational studies, this rule performed better than the two-pass Montagne's method which is a multi-pass procedure.

6. Computational Bench-Marks

We have so far compared the myopic heuristic against other competing heuristics and showed that it performed better than others in our computational study. However, in order to determine the efficiency of the myopic heuristic, it is necessary to find a high computational benchmark, if not the optimum. Some of the procedures used by the researchers for high computational benchmark are either to look for a tight lower bound which can easily be estimated or resort to truncated branch and bound(in truncated branch and bound the procedure is terminated after a finite number of nodes are evaluated and the best available upper bound is chosen for comparison purposes). The former procedure was used by Baker and Martin [1] in their study on the comparison of various heuristics for the weighted completion time problems for identical parallel processors and the latter procedure was used by Morton and Dharan [12] in evaluating the myopic heuristics for the weighted completion time problems in case of jobs with precedence constraints. Taking into consideration the fact that in an optimum solution no machine is idle(n > m), it can be shown that the total number of possible schedules is (n! n−1!)/(n−m! m−1!). Since there do not appear to exist any exact procedures which can find optimal solutions for reasonable size problems, we attempted to seek effective lower bounds.

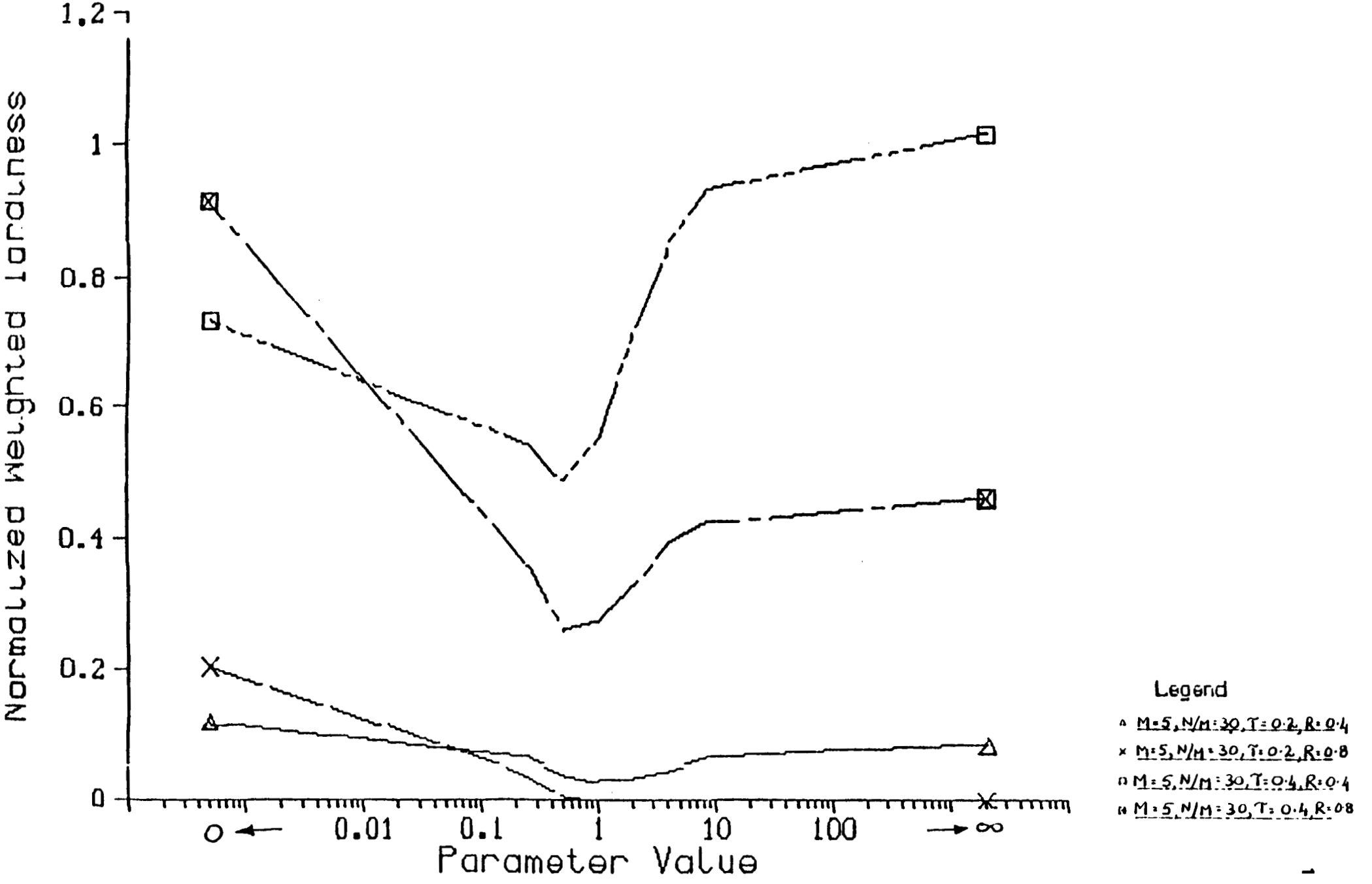
6.1 Lower Bound based on Weighted Lateness

It is clear that for any feasible schedule, weighted lateness is less than the weighted tardiness-

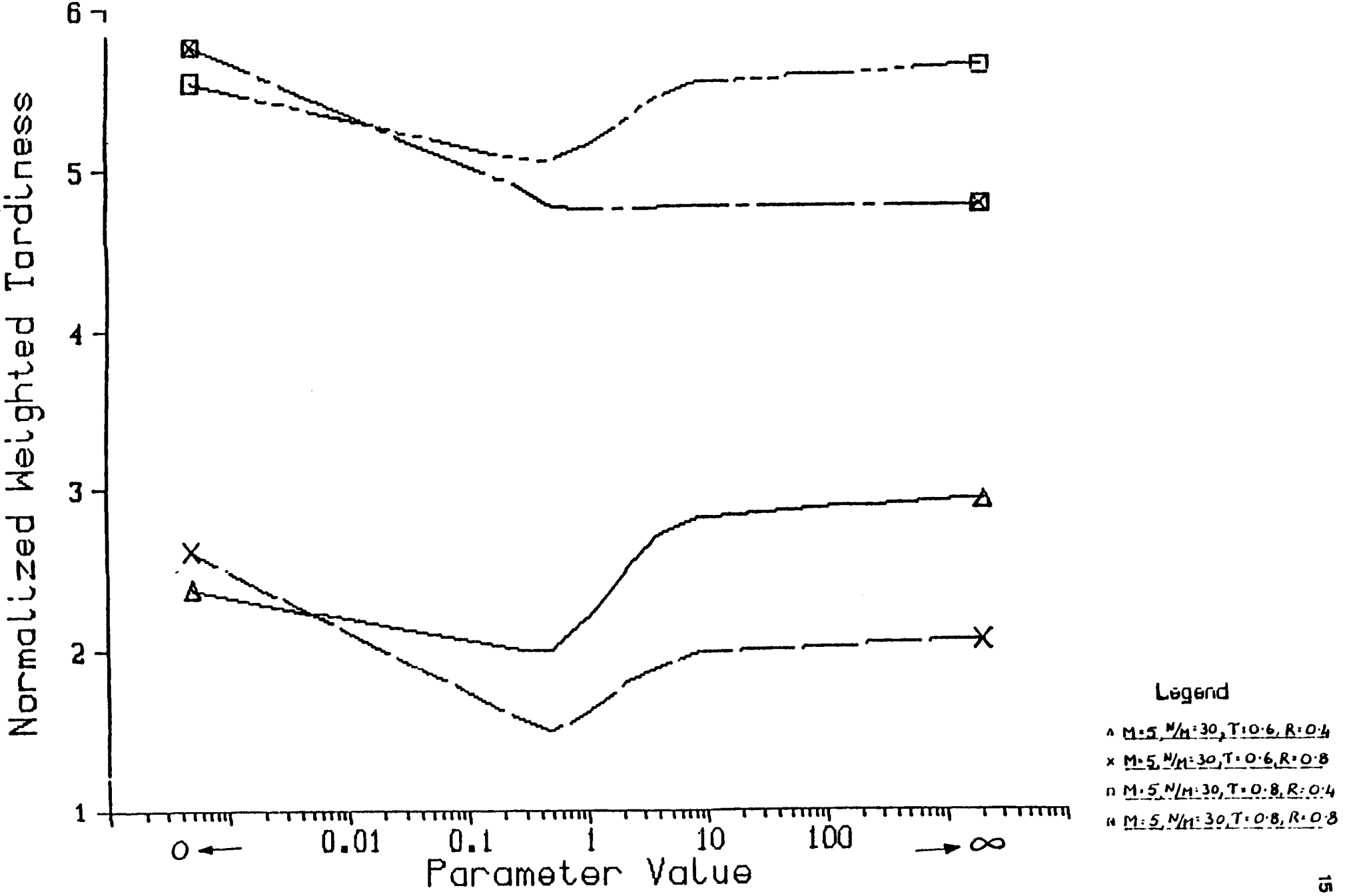
$$\Sigma_{i=1}^{i=n} w_i(c_i - d_i) \leq \Sigma_{i=1}^{i=n} w_i(c_i - d_i)^+$$

Hence a lower bound on the weighted lateness problem is a lower bound on the weighted tardiness problem also. Let LB1 be a lower bound on the weighted lateness

IGURE II- Performance of the myopic heuristic VS. Parameter Value



GURE III - Performance of the myopic heuristic VS. Parameter Value



problem and WL^* and WT^* be the values of optimal solutions to the weighted lateness and weighted tardiness problems respectively.

$$LB1 \leq WL^* \leq WT^*$$

Obviously, LB1 is rather a poor lower bound for the weighted tardiness problems when jobs have low tardiness. However, when the jobs have high tardiness, then a tight lower bound on the weighted lateness problems is likely to turn out to be an effective lower bound on the weighted tardiness problem as well(in the extreme case where all jobs are late in an optimum solution to the weighted tardiness problem, values of optimal solutions for both problems are the same).

Eastman, Even and Isaacs [6] have shown that the following is a lower bound for the weighted completion time problem in the case of identical parallel processors–

$$B(m) = \max \left\{ B(n),\ (1/m)B(1) + ((m-1)/2m)B(n) \right\}$$

where B(m) is the lower bound for the m processor problem. Using this bound,

$$LB1 \leq WL^* \Rightarrow LB1 \leq WC^* - \sum_{i=1}^{i=n} w_i d_i$$

$$LB1 \leq \max \left\{ B(n),\ (1/m)B(1) + ((m-1)/2m)B(n) \right\} - \sum_{i=1}^{i=n} w_i d_i$$

Since weighted tardiness is always non–negative,

$$LB1 = \max \left\{ 0,\ \max \left[B(n),\ (1/m)B(1) + ((m-1)/2m)B(n) \right] - \sum_{i=1}^{i=n} w_i d_i \right\}$$

We wish to further emphasize that this bound may be expected to be effective for problems with high values of tardiness factor.

6.2 Lower Bounds based on Pre-emptive Delivery

Suppose that the jobs can be pre–empted at unit intervals i.e., jobs can be split into pieces of unit length and can be delivered when finished. This relaxed problem can be formulated as the following linear programming problem–

$$\text{minimize } \sum_{i=1}^{i=n} \sum_{t=1}^{t=T} F_{it} x_{it}$$

$$\Sigma_{t=1}^{t=T} x_{it} \geq p_i \qquad \forall\ i \in N$$

$$\Sigma_{i=1}^{i=n} x_{it} \leq m \qquad \forall\ t \in T$$

$$x_{it} \leq 1 \qquad \forall\ i \in N,\ t \in T$$

$$x_{it} \geq 0 \qquad \forall\ i \in N,\ t \in T$$

where F_{it} equals $(w_i/p_i)(t-d_i)^+$ and T equals max { p_{max}, [np/m]+1 }.

It can easily be shown that the solution to the above problem is a valid relaxation for the original problem. It may also be noted that this problem can be solved using any of the available codes for solving capacitated transportation problems. Let LB2 be the optimal solution to this problem.

6.3 Lower Bounds based on Lawler's Procedure

Lawler [10] formulated the problem of scheduling jobs on identical parallel processors as a capacitated transportation problem. Adapting this procedure, the lower bound for the original problem can be found by solving the following problem–

$$\text{minimize } \Sigma_{i=1}^{i=n} \Sigma_{t=1}^{t=T} F_{it} x_{it}$$

$$\Sigma_{t=1}^{t=T} x_{it} \geq p_i \qquad \forall\ i \in N$$

$$\Sigma_{i=1}^{i=n} x_{it} \leq m \qquad \forall\ t \in T$$

$$x_{it} \leq 1 \qquad \forall\ i \in N,\ t \in T$$

$$x_{it} \geq 0 \qquad \forall\ i \in N,\ t \in T$$

where T is same as in LB2 and F_{it} is evaluated as follows:

$$F_{it} = 0 \qquad t \le 0$$
$$= kw_i \qquad d_i+(k-1)p_i+1 \le t \le d_i+kp_i$$

It can easily be shown that the above is a relaxation of the original problem. Let LB3 be the value of the optimal solution to this problem.

PROPOSITION III: LB2 ≤ LB3

PROOF: Without loss of generality, the job due dates can be reset to zero if they are negative. Also note that the feasible regions for LB2 and LB3 are the same. It is also clear that in any optimal solution to the problems, x_{it} take the values either 0 or 1.

Case I : Suppose $t = kp_i$, k an integer ≥ 0.

Contribution to the objective function in LB2= $(w_i/p_i)(kp_i)=kw_i$.

Contribution to the objective function in LB3= kw_i (by definition)

Therefore, LB2 = LB3

Case II: Suppose $t = d_i + (k-1)p_i + l$, $1 \le l \le p-1$, k an integer ≥ 0.

Contribution to the objective function in LB2

$$= (w_i/p_i)(t-d_i)^+$$
$$= (w_i/p_i)\{(k-1) + l/p_i\}^+ \qquad \text{----------} \qquad (1)$$

Contribution to the objective function in LB3

$$= kw_i \qquad \text{----------} \qquad (2)$$

Comparing (1) and (2), LB2 < LB3

Considering both cases I and II, LB2 ≤ LB3 ∎

From proposition III, it is clear that we do not need to compute LB2. However, LB1 is rather easy to compute. The size of Capacitated Transportation problems

described in §6.2 and 6.3 is a function of the number of machines and the processing times of the jobs. Due to problem size limitations, we formulated and solved additional 320 problems in the tardiness ranges 0.2,0.4,0.6 and 0.8. As was to be expected, LB1 was extremely weak, in most cases being 0, in the tardiness ranges 0.2,0.4 and 0.6. The additional 320 problems were run for the following set of parameters–

- number of machines: 2,4
- number of jobs per machine:
  - 15 jobs per machine and 20 jobs per machine in the case of 2 machines
  - 10 jobs per machine and 14 jobs per machine in the case of 4 machines
- tardiness factor: 0.2,0.4,0.6 and 0.8
- range factor: 0.4 and 0.8
- coefficient of variation for processing times: 0.1 and 0.3

Due to computational limitations, LB3 for the cases n=2,n/m=20 and n=4,n/m=14 could not be computed exactly. LB3 was computed for these cases by further relaxing the problem based on the following remarks–

REMARK I: If p_i is odd, then the problem solved with $p_i \leftarrow p_i - 1$ is a relaxation of the original problem.

REMARK II: If p_i is reduced to $p_i - 1$, then $d_i \leftarrow d_i - 1$ is a relaxation of the original problem.

REMARK III: If d_i is odd, then $d_i \leftarrow d_i + 1$ is a relaxation of the original problem.

However, problems solved making use of the remarks I, II and III tend to give weaker lower bounds. LB3 was determined by adapting the NETFLOW code devised by Kennington and Helgason [9]. The results comparing the performance of the myopic

TABLE V

Mean Deviation form the best lower bound
for the dispatch version of the myopic heuristic

Tardiness Factor = 0.2

		R=0.4		R=0.8	
n	n/m	Deviation from lower bound	Best lower bound	Deviation from lower bound	Best lower bound
2	15	0.0119	0.0113	0.0069	0.0029
2	20	0.0225	0.0229	0.0051	0.0009
4	10	0.0184	0.0187	0.0128	0.0029
4	14	0.0231	0.0138	0.0063	0.0000

Tardiness Factor = 0.4

2	15	0.0493	0.2603	0.0637	0.1621
2	20	0.1038	0.2742	0.1096	0.1514
4	10	0.0416	0.1660	0.0582	0.0755
4	14	0.0679	0.1758	0.0662	0.0690

Tardiness Factor = 0.6

2	15	0.1012	0.9009	0.1263	0.8226
2	20	0.2812	1.2383	0.2363	0.9586
4	10	0.0933	0.7628	0.1007	0.5444
4	14	0.1616	0.8224	0.1636	0.6652

Tardiness Factor = 0.8

2	15	0.2083	2.7561	0.1867	2.5254
2	20	0.3547	3.2259	0.2576	3.0736*
4	10	0.1053	1.8846*	0.1394	1.8294*
4	14	0.2440	2.2868*	0.1668	2.3890*

Unless otherwise stated, lower bounds reported are Lawler's lower bound. Lower bounds in case of problem sets marked with * are the lower bounds attained using Even's procedure.

TABLE VI

Mean Deviation of the myopic heuristic after adjacent pairwise interchange from the best lower bound

Tardiness Factor = 0.2

		R=0.4		R=0.8	
n	n/m	Deviation from lower bound	Best lower bound	Deviation from lower bound	Best lower bound
2	15	0.0084	0.0113	0.0053	0.0029
2	20	0.0204	0.0229	0.0035	0.0009
4	10	0.0158	0.0187	0.0179	0.0029
4	14	0.0200	0.0138	0.0038	0.0000

Tardiness Factor = 0.4

2	15	0.0457	0.2603	0.0600	0.1621
2	20	0.1007	0.2742	0.1025	0.1514
4	10	0.0392	0.1660	0.0480	0.0755
4	14	0.0621	0.1758	0.0555	0.0690

Tardiness Factor = 0.6

2	15	0.0995	0.9009	0.1217	0.8226
2	20	0.2808	1.2383	0.2328	0.9586
4	10	0.0925	0.7628	0.0962	0.5444
4	14	0.1601	0.8224	0.1613	0.6652

Tardiness Factor = 0.8

2	15	0.2079	2.7561	0.1867	2.5254
2	20	0.3530	3.2259	0.2566	3.0736*
4	10	0.1052	1.8846*	0.1383	1.8294*
4	14	0.2435	2.2868*	0.1658	2.3890*

Unless otherwise stated, lower bounds reported are Lawler's lower bound (LB3). Lower bounds in case or problem sets marked with * are the lower bounds attained using Even's procedure (LB1).

heuristic(with parameter value set at 1.0) against the best lower bound are shown in tables V. It may be noted that LB1 is a lower bound on the weighted lateness and LB3 is a lower bound on the weighted tardiness. Surprisingly, for high tardiness problems(τ=0.8), LB1 was tighter than LB3. Hence the lower bound in those cases is conservative. From tables V it is clear that the results are comparable to those obtained in the case of single machine problems(see [14]). We further attempted to improve the results of this heuristic by adjacent pairwise interchange. These results are shown in Tables VI. Comparing the results in tables V and VI, it is also clear that the gains through adjacent pairwise interchange are marginal.

7. Conclusion

It is clear from our study that the myopic heuristic performed better than the competing heuristics in the case of identical parallel machines. The advantage of the myopic heuristic is that it is simple to use and it can be easily incorporated into problem situations where other operational constraints may exist. Moreover, the myopic heuristic is a dispatch procedure which makes it easy to implement in practice. Our computational study has shown the efficacy of the heuristic by comparing it against a tight lower bound and further showed that an improvement routine such as adjacent pairwise interchange yields only marginal improvements. The procedure can easily be extended to the case of unequal parallel machines. The myopic heuristic can also be extended to flowshops. In case of flowshops, our heuristic can be used in dispatch mode at each machine with appropriate modifications. We are currently exploring this area and the preliminary results appear to be promising.

References

1. Baker, K.R., and Martin, J.B. "An experimental comparison of solution algorithms for the single machine tardiness problem." *Naval Research Logistics Quarterly 21*, 1 (January 1974), 187-199.

2. Barnes, J.W. and Brennan, J.J. "An Improved Algorithm for Scheduling Jobs on Identical Machines." *AIIE Transactions 9* (1977), 25-31.

3. Bernardo, J.J. and Lin, K.S. Scheduling independent jobs on Parallel Processors. Research paper

4. De,P.D., and Morton, T.E., "Scheduling to minimize Makespan on Unequal Parallel Processors." *Decision Sciences 11*, 4 (October 1980), 586-602.

5. Dogramaci, A. and Sukris, J. "Limitations of a parallel processor scheduling algorithm." *International Journal of Production Research 16*, 1 (1978), 83-85.

6. Eastman, W.L. "Comments on a paper by Schild and Fredman." *Management Science 11* (1965), 754-755.

7. Elmaghraby, S.E., and Park, S. "On the Scheduling of Jobs on a number of Identical Machines." *AIIE Transactions 6* (1974), 1-13.

8. Graves, S.C. "A Review of Production Scheduling." *Operations Research 29*, 4 (July-August 1981), 646-675.

9. Kennington, J.L., and Helgason, R.V., Algorithms for Network Programming.

10. Lawler, E. "On scheduling problems with deferral costs." *Management Science 11*, 2 (October 1964), 280-288.

11. Montagne, E.R., Jr. "Sequencing with time delay costs." *Arizona State University Industrial Engineering Research Bulletin* (January 1969), 20-31.

12. Morton, T.E., and Dharan, B.G., "Algoristics for Single Machine Sequencing with Precedence Constraints." *Management Science 24*, 10 (June 1978), 1011-1020.

13. Nunnikhoven, T.S. and Emmons, H. "Scheduling on Parallel Machines to Minimize two criteria related to Job Tardiness." *AIIE Transctions 19* (1977), 288-296.

14. Rachamadugu, R.V., and Morton, T.E., Myopic Heristics for the Single machine Weighted Tardiness Problem. Working Paper# 28-81-82, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh

15. Rajaraman, M.K. "Algorithm for scheduling Parallel processors." *International Journal of Production Research 13*, 5 (1975), 479-486.

16. Root, J.G. "Scheduling with Deadlines and Loss Functions on k Parallel machines." *Management Science 11*, 3 (January 1965).

17. Srinivasan, V. "A hybrid algorithm for the one machine sequencing problem to minimize total tardiness." *Naval Research Logistics Quarterly 18* (September 1971), 317-327.