Myopic Heuristics for the Single Machine Weighted Tardiness Problem

Thomas E. Morton and Ram Mohan V. Rachamadugu

CMU-RI-TR-83-9

Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

University of Michigan
Ann Arbor, Michigan

28 November 1982

This research was supported, in part, by the Air Force Office of Scientific Research under contract F49620-82-K0017.

Abstract

It is well known that the single machine weighted tardiness problem ($n/1//\Sigma w_i T_i$) is NP-complete. Hence, it is unlikely that there exist polynomially bounded algorithms to solve this problem. Further, the problem is of great practical significance. We develop myopic heuristics for this problem; these heuristics have been tested against competing heuristics, against a tight lower bound, and where practical, against the optimum, with uniformly good results. Also, these heuristics can be used as dispatching rules in practical situations. In our efforts to seek optimum solutions we develop a hybrid dynamic programming procedure (a modified version of Baker's procedure) which provides lower and upper bounds when it becomes impractical to find the optimum solution. Further, stopping rules are developed for identifying optimal first job/jobs.

MYOPIC HEURISTICS FOR THE SINGLE MACHINE WEIGHTED TARDINESS PROBLEM

1. Introduction

The problem of minimizing weighted tardiness of a given set of jobs to be processed on a single machine has attracted the attention of several researchers. Lenstra [9] has shown that the problem is NP−complete. In view of this, it is not surprising that earlier attempts in solving the problem resorted to both enumerative techniques and heuristics. Panwalkar, Dudek and Smith [7] report that in a survey conducted by them, the proportion of respondents who ranked meeting due dates or minimizing penalty costs as the most important criterion was larger than for any other criterion. In view of the practical importance of this problem,there a exists need for developing 'good' heuristics which are useful for the single machine case and may be extended and generalized to multiprocessors, flow shops and job shops.

Surprisingly, there are very few heuristics for the weighted tardiness problem. The problem may be defined as follows: we have n jobs $J_1, J_2, J_3,......J_n$ that arrive simultaneously to be processed on the machine. Each job J_i has associated with it a triple(p_i,d_i,w_i) which represents the processing time, the due date and the weight of the jobs. 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[1]

$$C_i(t_i) = w_i(t_i-d_i)^+$$

We wish to find a schedule such that $\Sigma_{i=1}^{i=n} C_i(t_i)$ is a minimum. Without loss of generality, we further assume that $d_i < \Sigma_{j=1}^{j=n} p_j$. Any job(s) not satisfying this condition can be deleted from the problem since there always exist optimal solutions in which

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

such a job(s) occupy the last position in the sequence. This condition can recursively be applied on the problem until the condition is satisfied.

2. Review of earlier heuristics

It is well known that if no job can be completed earlier than its due date, then the weighted shortest processing time rule(WSPT) minimizes weighted tardiness [1]. This is likely to be approximately the case when the machine or the shop is 'heavily loaded'.

Another heuristic which may be used is the earliest due date rule(EDD). Arrange the jobs according to the EDD rule. If it is possible under any rule to schedule all jobs on time, then the rule is optimal. This rule is likely to perform well when the shop or the machine is 'lightly loaded' [13].

Taking into consideration the fact that these simple heuristics perform well under these extreme situations, Schild and Fredman [13] developed a procedure that they claimed to give an optimal schedule. However, Eastman [6] showed that the procedure is not an exact one by constructing a counterexample. No computational studies have been reported to determine how good a solution is generated by their procedure.

In a paper on the experimental comparison of solution algorithms for the average(unweighted) tardiness problems, Baker and Martin [1] refer to Montagne's method [10]. 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 $p_j/w_j(\Sigma_{i=1}^{i=n}p_i-d_j)$ [3].

Yet another heuristic proposed by Baker [4] for the average or unweighted tardiness problem, called 'modified due date method', is as follows: if it is impossible to complete a job before its due date revise its due date to be the earliest possible completion time. Schedule next the job that has the earliest due date. It appears that the procedure has done well in experimental studies [4]. It can easily be seen that Baker's rule indeed provides optimal solution in two extreme cases for the unweighted or average tardiness problems– when all jobs in an optimal sequence are either early or late.

3. Description of our heuristic

Prior to the description of our heuristic, consider the following property which characterizes an optimal solution to the single machine weighted tardiness problem.

PROPOSITION I: Let J_i and J_j be any two adjacent jobs (J_i precedes J_j) in an optimal sequence for the single machine problem. The sequence satisfies the following property–

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

where t is the start time for J_i

PROOF: We have to consider six subcases. These are as follows:

Case I: Both jobs are early in either position(Figure 1). In this case we are indifferent as to which sequence(J_i immediately precedes J_j or J_j immediately precedes J_i) is used. If J_i does not precede J_j in a given optimal sequence, we can create another optimal sequence satisfying the property by merely interchanging jobs J_i and J_j.

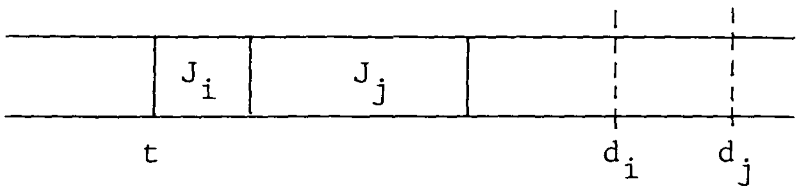

Figure 1

Case II: Both jobs are late in either position(Figure 2).

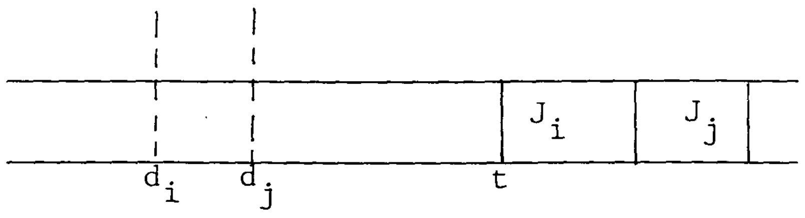

Figure 2

Since both jobs are late in either position, it is necessary that the the job with higher ratio of the weight to the processing time must be scheduled first for the sequence to be optimal. Since $d_i<t+p_i$ and $d_j<t+p_j$,

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

Case III: One job is late in either position and the other is early in the earlier position and late in the later position(Figure 3)

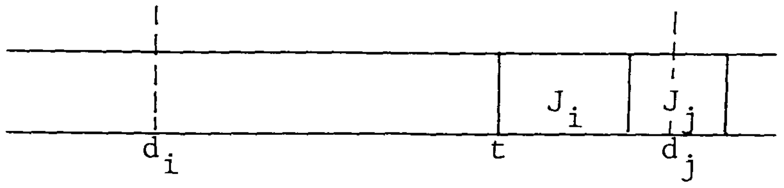

Figure 3

$d_j > t + p_j$ $\qquad$ $d_j < t + p_i + p_j$ $\qquad$ $d_i < t$

Cost if J_i precedes J_j = $w_i(t+p_i-d_i) + w_j(t+p_i+p_j-d_j)$

Cost if J_j precedes J_i = $w_i(t+p_i+p_j-d_i)$

J_i should precede J_j if

$w_i(t+p_i+p_j-d_i) \geq w_i(t+p_i-d_i) + w_j(t+p_i+p_j-d_j)$

$$\frac{w_i}{p_i} \geq \frac{w_j}{p_j}\left\{1 - \frac{(d_j - t - p_j)}{p_i}\right\}$$

Since $d_i < t$ and $d_j < t + p_i + p_j$, the above expression may be rewritten as

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

Case IV: One job is late in either position and the other is early in either position(Figure 4).

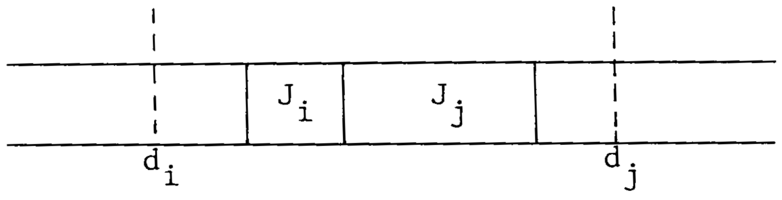

Figure 4

$d_i < t$ $\qquad$ $d_j > t+p_i+p_j$

It is obvious that J_i should precede J_j

Since $d_j-(t+p_j) > p_i$, $\quad \frac{w_j}{p_j}\left\{1 - \frac{(d_j - t - p_j)^+}{p_i}\right\}^+ = 0$

Since $d_i < t$,

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

Case V: One job is early in either position and the other is early in the earlier position and late in the later position(Figure 5).

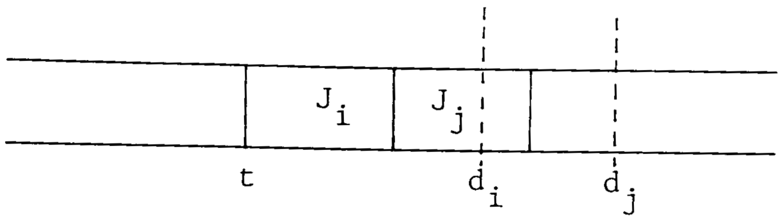

Figure 5

$d_j > t+p_i+p_j \quad d_i > t+p_i \quad d_i < t+p_i+p_j$

It is clear that in this case J_i should precede J_j.

Since $d_j-(t+p_j) > p_i$,

$$\frac{w_j}{p_j}\left\{1 - \frac{(d_j - t - p_j)^+}{p_i}\right\}^+ = 0$$

Since $w_i > 0$, $d_i-(t+p_i) > 0$ and $d_i-(t+p_i) < p_j$,

$$\frac{w_i}{p_i}\left\{1 - \frac{(d_i - t - p_i)^+}{p_j}\right\}^+ \text{ is positive.}$$

Therefore,

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

Case VI: Both jobs are early in the earlier position and late in the later position(Figure 6).

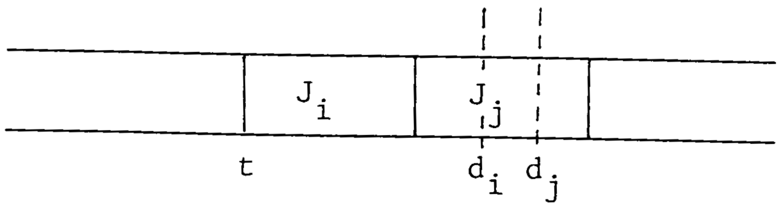

Figure 6

$d_i > t+p_i$ and $d_i < t+p_i+p_j$

$d_j > t+p_j$ and $d_j < t+p_i+p_j$

J_i should precede J_j if

$w_i(t+p_i+p_j-d_i) > w_j(t+p_i+p_j-d_j)$

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

Thus, in all cases the property is satisfied by at least one optimal solution. ■

This proposition can be used directly to find a schedule which cannot be improved by adjacent pairwise interchange. We exploit this property in the following manner in developing our heuristic: for every job, we determine an 'apparent priority index'(AP_i) as defined below:

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

where t is the current time. Since at any instance, we do not know what the optimal first two jobs on the machine would be, we approximate the value of p_j by X. In the absence of any estimate, we approximate the value of p_j by the mean processing time of the jobs. However, it may be noted that in assigning X value equal to the mean processing time of the jobs, we are in fact trying to strive towards local optimality. It is clear that since local optimality does not necessarily ensure global optimality in this problem, we may attempt to assign X a value which is more than one multiple of the average processing time of the jobs, thus helping us look beyond the next job and achieve better results.

Our heuristic is as follows: at any instance, we determine the apparent priority for all unscheduled jobs. We assign next the job with the highest apparent priority. In case of ties, we assign next the job that has the earliest due date(the secondary criterion is based on our study of a relaxation of the problem where all jobs have equal processing times and equal weights. It is also interesting to note the existence of a property similar to the one we discussed for the relaxed problem with jobs having equal processing times. In this case, the result holds good not only in the case of adjacent pairwise interchange, but also when comparing jobs not necessarily adjacent to each other in an optimal solution. These details are presented in the appendix).

It is interesting to note the change in apparent priority assigned by our heuristic over time. This is shown in Figure 7. It is clear that if a job is too early, then it need not be scheduled immediately. Also, if the job is late, it is given full priority(w_i/p_i) as in WSPT rule. In the intermidiate range, the apparent priority is smoothly increased. Also, we note that as $X \rightarrow \infty$, our heuristic is same as WSPT rule. However, as $X \rightarrow 0$, it assignec priority as follows:

$$\begin{aligned} AP_i &= 0 && \text{if slack is positive} \\ &= w_i/p_i && \text{if slack is zero or negative} \end{aligned}$$

When we impose the secondary priority rule also, it may be noted that as $X \rightarrow 0$, our heuristic behaves somewhat like EDD rule, but not quite the same. However, even when jobs are rather slack, our heuristic appears to have performed better than the EDD rule(see the section on computational experiments).

An appropriate choice of X is necessary for the good performance of our heuristic. Intuitively, as explained before, one would expect it to be related to the average processng time of the jobs. So the apparent priority may be written as follows:

$$\text{H1:} \quad AP_i = \frac{w_i}{p_i} \left\{ 1 - \frac{(d_i - t - p_i)^+}{k\bar{p}} \right\}^+$$

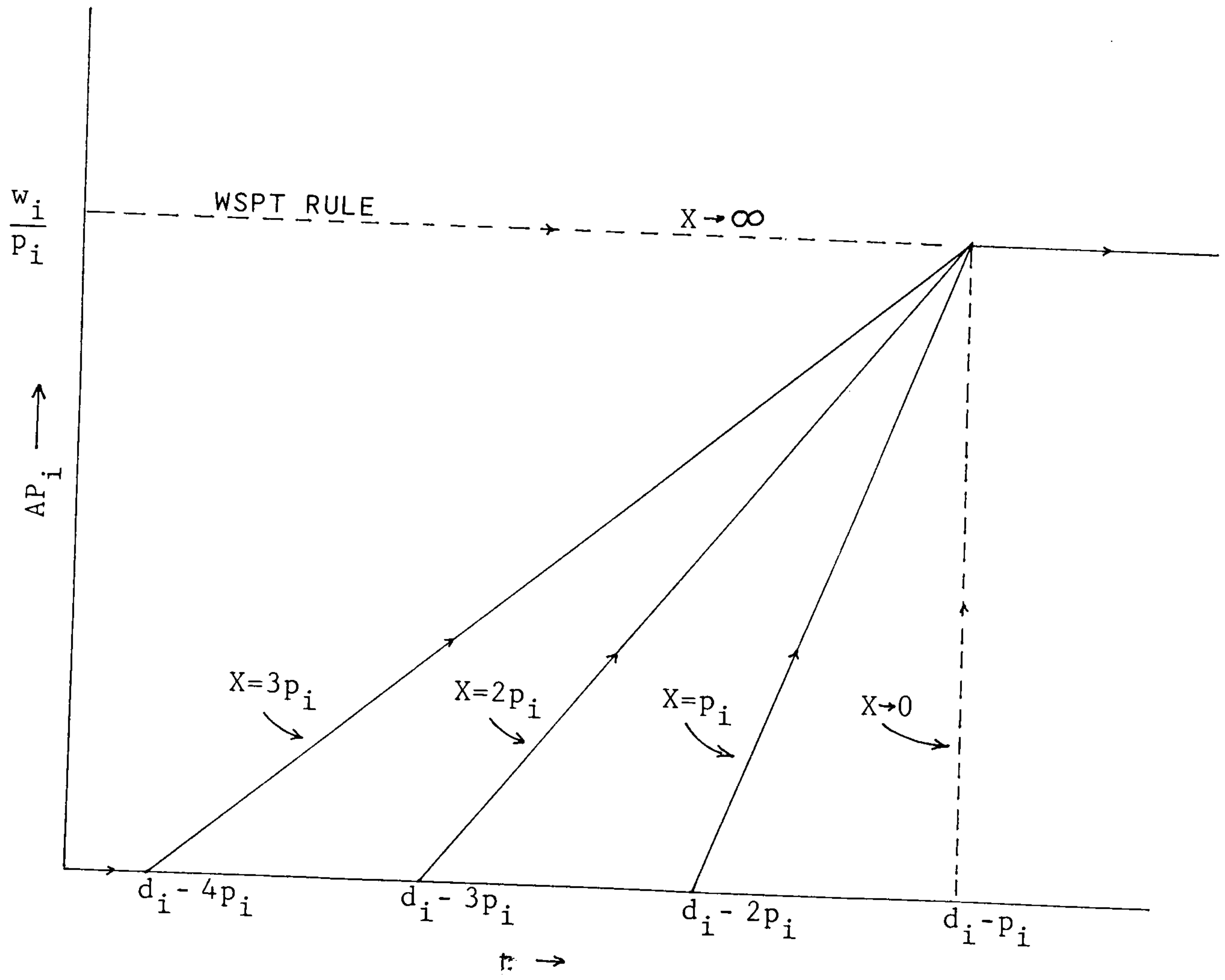
$\frac{w_i}{p_i}$
WSPT RULE
$X \to \infty$
$AP_i \longrightarrow$
$X=3p_i$
$X=2p_i$
$X=p_i$
$X \to 0$
$d_i - 4p_i$
$d_i - 3p_i$
$d_i - 2p_i$
$d_i - p_i$
$t \rightarrow$

FIGURE 7

where k is a parameter to be determined and $\bar{p}$ is the average processing time of unscheduled jobs. It is possible for us to develop different rules for assigning apparent priority for the jobs. However, we would expect these alternate schemes to have features similar to H1 such as assigning the job full priority once(w_i/p_i) it is late and zero or near zero priority if it is too early. In the intermediate range, we may follow alternate schemes which gradually increase the priority of the job. Two alternate scemes, where the rate of change in the priority of the job in the intermediate range itself increases over time are envisaged below:

$$\text{H2:} \quad AP_i = \frac{w_i}{p_i}\left\{1 - \frac{\bar{p}}{\bar{p} + k(d_i - t - p_i)^+}\right\}$$

$$\text{H3:} \quad AP_i = \frac{w_i}{p_i} \exp\left(- \frac{k}{\bar{p}}(d_i - t - p_i)^+\right)$$

H2 and H3 are similar to H1. Their characteristics are shown in Figures 8 and 9 respectively. It may be noted that in these cases, as in H1, jobs are assigned full priority(w_i/p_i) if the slack is zero or negative. However, as is evident from Figures 8 and 9, rate of change in the priority assigned to a job increases as t is increased until there is no more slack. In our pilot studies, we found that H3 performed better than H1 and a parameter value of k in the range of 0.5 to 2 yielded good results over wide range o·· problems.

It is also interesting to note the asymptotic forms of the heursistics. These are shown in table 1.

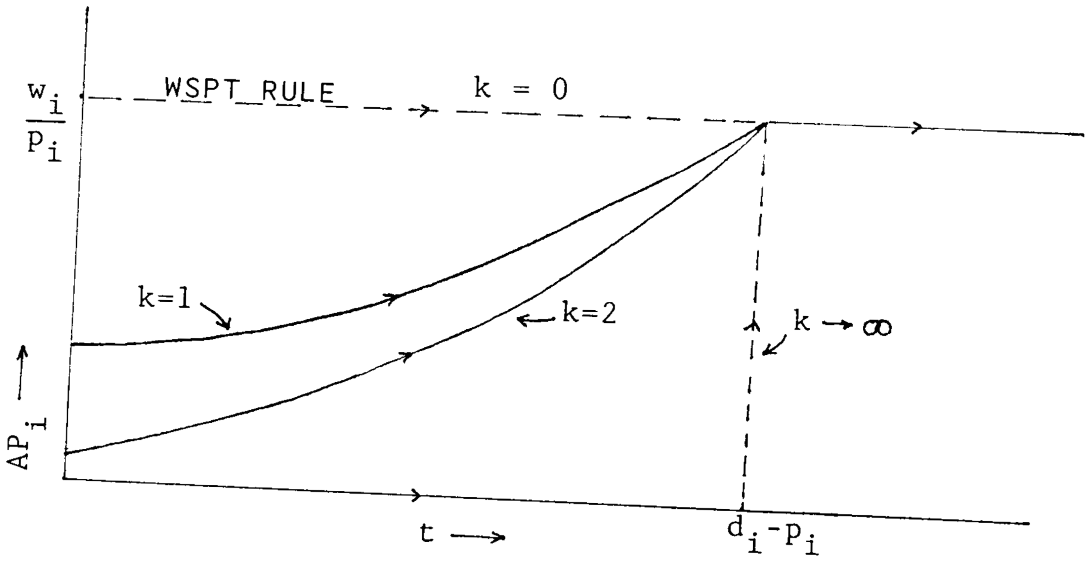

FIGURE 8

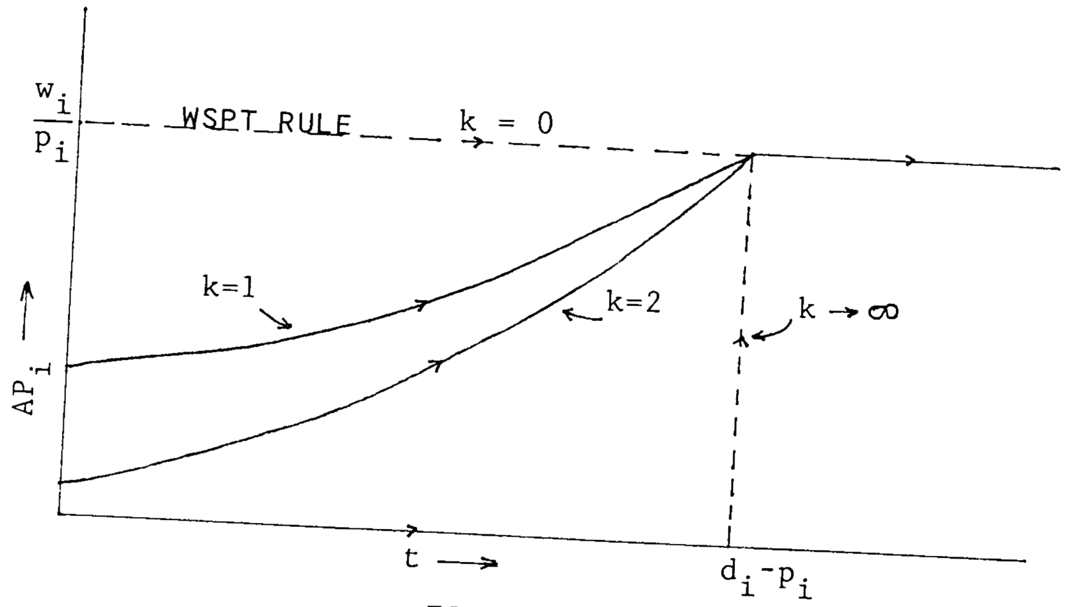

FIGURE 9

	Apparent Priority	
Heuristic ↓	$k = 0$	$k \to \infty$
H1	0 if early $\frac{w_i}{p_i}$ o/w	Same as WSPT Rule
H2, H3	Same as WSPT Rule	0 if early $\frac{w_i}{p_i}$ o/w

Table 1

4. Review of prior computational studies

In testing out various enumerative algorithms for the weighted tardiness problem (and also unweighted or average tardiness problem), various authors followed different procedures for generating test problems. [2]

Two important factors over which control was exercised in generating test problems are the tardiness factor and the due date range. In most prior studies, it was assumed that the job weights were independent of other factors. The tardiness factor is a rough measure of the number of jobs which might be expected to be tardy in a random sequence [16]. Let $\bar{p}$ be the mean processing time and $\bar{d}$ be the average due date. Then, in an average sense, the number of jobs completed in time in a random sequence is given by d/p. The tardiness factor, τ, is given by

$$\begin{aligned} \tau &= 1-\text{Proportion of jobs on time} \\ &= 1-((\bar{d}/\bar{p})/n) \end{aligned}$$

$$\bar{d} = n\bar{p}(1-\tau)$$

The typical procedure followed by various authors in generating the test problems is as follows: generate the p_i as per some distribution and generate the due dates using the tardiness factor and population mean or the sample mean of the processing times. The range for the due dates was controlled by specifying the variance of the distribution generating the due dates.

Srinivasan [16], in testing his hybrid algorithm for the average or unweighted tardiness problem used a bivariate normal distribution for generating processing times and the due dates. Srinivasan generated test problems controlling for the following factors: the coefficient of variation for the processing times, the coefficient of variation for the due dates, the correlation coefficient between the processing times and the due dates. The number of jobs in a problem was varied from 8 to 50. His results indicated that the problems with tardiness factor of 0.6 were most difficult to solve.

In a study comparing the effectiveness of various algorithms for unweighted or average tardiness problem, Baker and Martin [1] followed a similar procedure, but used a normal distribution to generate processing times and uniform distribution to generate due dates. The range of the due dates was varied from 20% to 95% of the total processing times of the jobs. The number of jobs in a problem was varied from 8 to 15.

Fisher [8], in testing a dual based procedure for solving average or unweighted tardiness problem, used a uniform distribution to generate both the processing times and the due dates. He tested his procedure on problems with the number of jobs varying upto 50, tardiness factor varied from 0.5 to 0.8 and the range of the due dates varied from 20 to 100% of the total processing time of the jobs. His conclusions regarding the problem difficulty are similar to those of Srinivasan [16].

Schweimer [15], in testing his branch and bound procedure for the weighted tardiness problem, generated processing times from a uniform distribution[1,10] and the due dates were. generated from a uniform distribution[p_i,5.5n]. Job weights were generated from a uniform distribution [1,5]. Number of jobs in a problem were chosen to be 10 or 20. It may be noted that the weights were generated independent of the processing times and the due dates. It may also be noted that no control was exercised over the tardiness factor. In fact, it can be shown that tardiness factor was implicitly set at approximately 0.5.

In a study conducted by RinnooyKan *et al* [12] to test their branch and bound algorithm for the weighted tardiness problem, weights were generated from a uniform distribution[4.5,15.5]. Problem sizes of 10,15 and 20 were tried. Tardiness factor was set at 0.2,0.4,0.6 and 0.8. Processing times were generated using the Normal distribution and the due dates were generated from a uniform distribution. As in Schweimer's study, job weights were generated independent of the processing times and the due dates. RinnoyKan *et al* study indicated no relation between computational time and the correlation coefficient between processing times and the due dates. Problems with large range for due dates were relatively easier to solve compared to problems with short range for the due dates. RinnooyKan *et al* study indicated that the problems with tardiness factor of 0.8 were difficult to solve(compared with 0.6 in Srinivasan's study [16]). However, any such comparison must take into consideration the fact that RinnooyKan *et al* study was on the weighted tardiness problem whereas Srinivasan's study was on the average or unweighted tardiness problems.

Picard and Queyrenne [11] tested their adaptation of time dependent travelling salesman algorithm to the weighted tardiness problem on the same set of problems used by RinnooyKan *et al*. Schrage and Baker [14] used the same set of problems generated by RinnooyKan *et al* to test their procedure.

5. Measure of performance

Prior computational studies on the weighted tardiness problem were largely confined to validating enumerative methods. This being the case, it is not surprising that the emphasis in these studies was on the use of computational time and/or memory requirements. However, in our study, we wish to find how 'good' our heuristic is when compared to the optimum value. Since this implies that the study is to be conducted across wide range of values of number of jobs in a problem,processing times of jobs, weights etc., the performance measure should take these aspects into consideration. Absolute deviation from the optimum value is likely to suffer from scaling effects. Any averaging of the percentage deviation from the optimum is likely to mislead us since such deviations are likely to be very large in the case of problems with low tardiness factor(For a more detailed discussion of the choice of appropriate measure of performance, see [5]). The metric that we will be using in our study is as follows:

$$\text{Performance of the heuristic:} \quad \frac{\text{Weighted tardiness for heuristic sequence}}{W * n * p} - \frac{\text{Optimum value}}{W * n * p}$$

W,n and p are, respectively, the mean weight of the jobs, number of jobs and the mean processing time of the jobs in a problem. We normalize the performance measure by dividing the deviation from the optimum by the number of jobs. This normalizes the measure with respect to the number of jobs in a problem and thus permits comparison among problems with different number of jobs. Further division with the average weight normalizes the measure for the differences in the average weights of the job sets in different problems. Finally, division with the average processing time expresses the measure in terms of the number of **average processing times tardy**.

In case of problems where the optimum value could not be found due to computational limitations such as time and/or memory requirements, we used a tight lower bound and the best feasible solution.

6. Method for obtaining optimum or 'high bench mark' solution

In order to test our heuristic, it is necessary that we compare the performance of our heuristic against the optimum, if possible. Based on the reported performance results, three enumerative methods [14, 11, 12] seem most promising. Of all the enumerative methods, we choose the dynamic programming procedure suggested by Schrage and Baker [14]. Among the various enumerative methods, this procedure has the best computational time performance for the set of tested problems. Furthermore, the labelling procedure used in this method leads to compact memory requirements, particularly in case of the problems with high tardiness value. These are the very problems that have been found by other researchers most difficult to solve. Also, the stopping rule that we develop for identifying first job/jobs in an optimal solution is based on the dynamic programming procedure.

It is however possible that, though the dynamic programming approach suggested by Baker and Schrage [14] requires the least computational time, labelling space requirement may be too large, particularly in case of the problems with low tardiness factor. These are the problems for which no computational results have been reported by Baker and Schrage. Also, none of the earlier studies have reported results for problems having more than 20 jobs in case of weighted tardiness problems. Since we planned to test problems having more than 20 jobs, it seemed likely that we might be constrained by limitations of excessive memory requirements and/or excessive computational time. In such cases, we compared the performance of our heuristic against a 'high bench-mark', such as a tight lower bound. Unfortunately, Schrage and Baker [14] procedure does not compute lower and upper bounds for the problem.

Since it is most likely that in case of large problems(problems with more than 20 jobs) we might be constrained by the limitations of computational time and/or memory requirements, we modified the Baker and Schrage procedure [14] to determine the lower and upper bounds. The procedure was further modified to arrange the jobs in stages, which was necessary to determine the lower bounds and also for the use of a stopping rule developed by us. The bounds become sharper and sharper as we progressively move from one stage to the next. The details of the hybrid dynamic programming procedure developed by us are shown in the next section.

6.1 Hybrid dynamic programming procedure

This procedure is a modification of the dynamic programming procedure for the sequencing problems with precedence constraints developed by Schrage and Baker [14]. We modified this procedure in order to determine the lower and upper bounds at every application of the recursive relationship. We also developed a stopping rule for identification of first job in an optimal sequence. We follow notation similar to Baker and Schrage [14] with appropriate additions as needed for our modification of the procedure.

Notation

J_i	: Job i
S	: set of feasible jobs. S is feasible if, for every job $J_i \in S$, all the predecessors of J_i are also included in S.
N	: Set of all jobs.
t(S)	: $\Sigma_{j \in S}\, p_j$
$\bar{S}$	: $N \setminus S$
f(S)	: Value of the optimal schedule for set S
R(S)	: Set of jobs in S that have no successors in S
g(k,t(S))	: Penalty for completing J_k at t(S), $k \in S$
WSPT($\bar{S}$)	: Value of minimum weighted lateness schedule for the jobs in $\bar{S}$ with the release date being t(S)
B(S)	: Lower bound for the weighted tardiness problem given that feasible set S is scheduled optimally at the beginning
F(S)	: Index of the job scheduled to be in the first position in the sequence generated for f(S)
LB(l)	: Lower bound for the problem given that all feasible subsets of cardinality l have been enumerated.

Recursive relation is [16],

$$f(S) = \min_{k \in R(S)} \{ f(S \setminus k) + g(k,\ t(S)) \}$$

Initial condition is $f(0)=0$
Optimal value is given by $f(N)$.

Schrage and Baker [14] provided the detailed procedure for enumerating all feasible subset S in such a way that $S\backslash k$ is enumerted before S and a procedure for assigning an address to the subset $S\backslash k$ so that $f(S\backslash k)$ can be accessed quickly.

At every enumeration, we determine B(S) as follows:

$$B(S) = f(S) + \max\{0, \text{WSPT}(\bar{S})\}$$

If $B(S) \geq$ current best feasible solution, then f(S) can be set at infinity and need not be further considered. Further, a lower bound for the problem is given by

$$LB(l) = \min B(S) \quad \forall \quad |S| = l \text{ and } S \subseteq N$$

An upper bound for the solution is given by

$$UB(S) = f(S) + \text{weighted tardiness of WSPT sequence for jobs in } \bar{S}$$

We terminate if $UB(S) = LB(|S| - 1)$

<u>**6.2 Stopping rule for the optimal first job**</u>

In order to guarantee the optimal first job, we can use the following procedure: suppose F(S) is same for all S such that $|S|=l$, $l=2,3,\ldots n$. Stop further computation after the condition is satisfied for the smallest value of l.

For identifying the optimal first job and/or determinig the lower bounds, it is necessary to know when all feasible subsets of jobs of given cardinality have been enumerated. This may be done by numbering the jobs and arranging the jobs in stages as shown below:

1. Jobs are assigned to stages such that no job is assigned to a stage less

than or equal to its predecessors.

2. Jobs at any stage have indices greater than jobs at earlier stages.

3. Every job is assigned to the earliest possible stage, subject to (1) and (2).

These details are shown for a hypothetical example in Figure 10. It may be noted that, when the above mentioned job indexing procedure is used in conjunction with the enumeration scheme proposed by Baker and Schrage [14], all feasible subsets of cardinality k−1 would have been enumerated before the job with the lowest index in stage k can be considered for inclusion in a feasible subset of tasks. Thus, the updating of LB(l) and checking for the optimal first job can be carried out when the job with the lowest index at any stage is being considered for the first time for inclusion in the feasible set S.

Another independent stopping rule for identifying the optima first job follows from the next proposition–

<u>PROPOSITION II</u>: If the job with the highest w_i/p_i is tardy even if scheduled first, then there is an optimal sequence in which it must be sequenced first.

<u>PROOF</u>: Without loss of generality, assume that $w_1/p_1 > w_2/\ p_2$ .... Also, since J_1 is tardy even if scheduled first, $p_1 > d_1$. Suppose there exists an optimal schedule such that J_1 occupies jth position and let J_i occupy j−l th position(Figure 11).

Pairwise interchange of J_i and J_1 does not affect the completion times of other jobs. Decrease in the value of the objective function due tc pairwise interchange of J_i and J_1 equals

$$w_i[\{0,T+p_i-d_i\}^+-\{0,T+p_i+p_1-d_1\}^+] + w_1[\{0,T+p_i+p-(1)-d_1\}^+-\{0,T+p_1-d_1\}^+]$$

$$\geq w_1p_i - w_ip_1$$

$$\geq (p_ip_1)^{-1}[(w_1/p_1) - (w_i/p_i)]$$

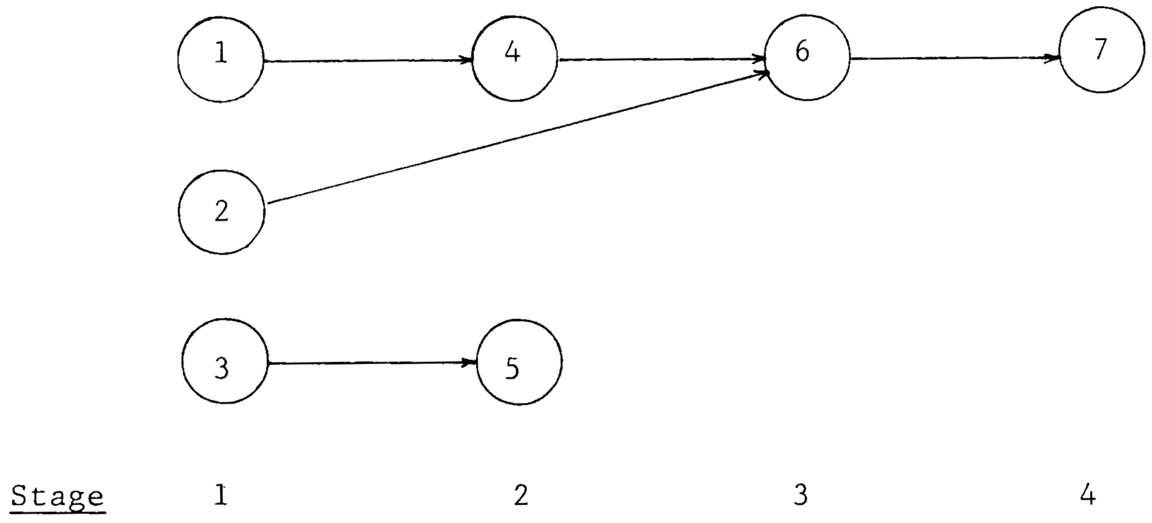

FIGURE 10

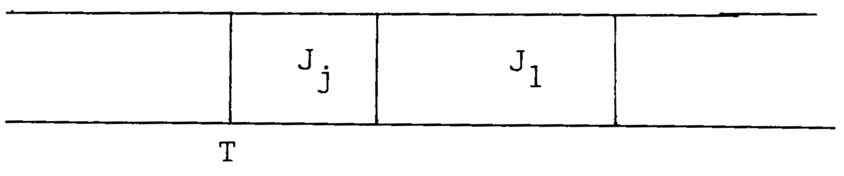

FIGURE 11

However, the right hand side is non negative and this contradicts the optimality of the original schedule. Thus, by successively 'pushing' J_1 to the first position, we get set of dominant schedules and hence the result.

7. Design of the experiment

Control variables in generating the test problems are: number of jobs in a problem, distribution of the processing times, distribution of the due dates, correlation between the processing times and the due dates, priority or the weights assigned to the jobs.

- Processing times and the due dates: Processing times and the due dates are generated using bivariate Normal distribution which incorporates the variation in processing times, variation in due dates and the correlation between the processing times and the due dates. We set the various parameters at the following levels:

Tardiness factor(τ)	:0.2,0.4,0.6,0.8
Coefficient of variation for the processing times	:0.1,0.3
Correlation coefficent between p_i and d_i (ρ)	:0,0.5
Range factor for the due dates (R)	:0.4,0.8
Population mean for the job processing times	:30

- Weights for the jobs: In prior studies by RinnooyKan [12] and Schweimer [15], job weights were generated independently of the job processing times and the due dates. However, we feel that on average the penalties associated with the tardiness of the jobs would be proportionate to the work content of the jobs. Taking this into consideration, we determine the weights for the jobs by independently determining the factor w_i/p_i from the uniform distribution in the range [0,2].

$$w_i=(w_i/p_i)''*p_i$$

$(w_i/p_i)''$ is random variate generated from the uniform distribution [0,2] and p_i is the processing time generated from a bivariate normal distribution as described above.

- Number of jobs: In order to study the effect of the number of jobs in a problem on our heuristic, we choose the number of jobs in a problem to be 10, 20 or 30.

We tested 20 problems for each specification of the parameters. Thus, in total we tested 20x4x2x2x2x3=1920 problems.

7.1 Computational experiments

In testing our heuristic(for comparison purposes, we used exponent form of our heuristic[H3] with parameter value set at 0.5) on 1920 problems, we made a few further changes. For problems where optimum solution could not be found(largely due to excessive memory requirement for problems with 30 jobs), we compared myopic heuristic solution against lower and upper bounds. We found additional lower and upper bounds by solving the linear assignment relaxation procedure suggested by RinnooyKan[2] *et al* [12]. Best upper bound for the solution was found by choosing the best solution among EDD sequence, WSPT sequence, Montagne's sequence, upper bound generated by the hybrid dynamic procedure at termination, solution to linear assignment relaxation procedure suggested by RinnooyKan *et al* and fifteen solutions generated by five parameter values for each of the three different versions of our heuristic.

Tables 2 through 5 give the computational results for various problem sizes. Table 2 provides the results for problems with 10 and 20 jobs. As may be noted, our heuristic performed well when compared to other heuristics. As noted earlier, we kept the parameter value of the myopic heuristic fixed at 0.5. However, results can

[2] Our pilot studies as well as published results [12] showed that the lower bound obtained by this procedure is about 20% below the optimum value. Howver, the lower bound tends to be tighter if the problems are less tardy and/or the variance of the job processing times is low.

TABLE 2

Mean Value of Performance Measure for 10 and 20 Job Problems

n	τ	R = 0.4					R = 0.8				
		OPT	EDD	WSPT	MP	MYH	OPT	EDD	WSPT	MP	MYH
10	0.2	0.038	0.770	0.047	0.014	0.020*	0.017	0.028	0.107	0.011	0.026
	0.4	0.253	0.515	0.094	0.048	0.029	0.184	0.293	0.232	0.084	0.051
	0.6	0.886	1.006	0.151	0.088	0.027	0.740	0.953	0.376	0.183	0.055
	0.8	2.090	1.427	0.112	0.065	0.015	2.094	1.402	0.202	0.088	0.025
20	0.2	0.024	0.107	0.074	0.027	0.021	0.007	0.024	0.151	0.022	0.014
	0.4	0.403	0.830	0.192	0.125	0.033	0.196	0.513	0.498	0.202	0.047
	0.6	1.319	1.981	0.298	0.194	0.035	1.128	1.697	0.774	0.398	0.054
	0.8**	3.619	2.897	0.337	0.219	0.018	3.513	0.018	3.064	0.587	0.220

OPT:	Mean Value of Normalized Optimum
EDD:	Earliest Due Date Rule
WSPT:	Weighted Shortest Processing Time Rule
MP:	Montagne's Procedure
MYH:	Myopic Heuristic [H3] with parameter k value set at 0.5

*Increasing the value of parameter k in the myopic heuristic yields better results than Montagne's method.

**One problem was not solved to optimality in case of both range factors - 0.4 and 0.8. However, myopic heuristic yielded the best feasible solution for both problems.

TABLE 3

Mean Value of Performance Measure for fully solved 30 Job Problems

(n = 30)

R = 0.4

τ	Number of problems fully solved	OPT	EDD	WSPT	MP	MYH
0.2	73	0.027	0.107	0.099	0.035	0.017
0.4	26	0.400	1.125	0.290	0.164	0.027
0.6	8	2.069	2.049	0.439	0.350	0.056
0.8	16	5.186	4.242	0.564	0.315	0.018

R = 0.8

τ	Number of problems fully solved	OPT	EDD	WSPT	MP	MYH
0.2	80	0.001	0.033	0.224	0.020	0.007
0.4	38	0.172	0.521	0.739	0.260	0.048
0.6	10	1.600	2.412	1.215	0.634	0.073
0.8	20	5.380	4.223	0.837	0.352	0.030

OPT:	Mean Value of Normalized Optimum
EDD:	Earliest Due Date Rule
WSPT:	Weighted Shortest Processing Time Rule
MP:	Montagne's Procedure
MYH	Myopic Heuristic [H3] with parameter k value set at 0.5

TABLE 4

Mean Value of Normalized deviation from the best lower bound
(For unsolved 30 Job Problems)

τ	R = 0.4					R = 0.8				
	Number of Problems	EDD	WSPT	MP	MYH	Number of Problems	EDD	WSPT	MP	MYH
0.2	7	0.413	0.196	0.150	0.091	0	-	-	-	-
0.4	54	1.457	0.514	0.414	0.250	42	1.157	0.853	0.481	0.209
0.6	72	3.586	1.140	0.940	0.713	70	3.165	1.793	1.188	0.664
0.8	64	5.098	1.056	0.876	0.592	60	4.799	1.452	0.923	0.487

EDD:	Earliest Due Date Rule
WSPT:	Weighted Shortest Processing Time Rule
MP:	Montagne's Procedure
MYH:	Myopic Heuristic [H3] with parameter k value set at 0.5

TABLE 5

Comparison of Mean Values for myopic heuristic vis-a-vis best lower and best upper bounds
(For unsolved 30 Job Problems)

τ	R = 0.4			R = 0.8		
	Best Normalized Lower Bound	Best Normalized Upper Bound	Normalized Myopic Heuristic Value	Best Normalized Lower Bound	Best Normalized Upper Bound	Normalized Myopic Heuristic Value
0.2	0.071	0.155	0.162	-	-	-
0.4	0.294	0.526	0.544	0.167	0.356	0.376
0.6	1.270	1.909	1.983	0.925	1.522	1.589
0.8	4.549	5.096	5.141	4.504	4.965	4.991

further be improved at low tardiness factors by increasing the value of the paramemter k. In case of problems with 20 jobs, we found optimum for all problems except two problems with tardiness factor 0.8.

In the case of 30 job problems, we could not find the optimal solution to all problems. Results comparing the performance of various heuristics for problems where optimum could be found are shown in Table 3. It is clear that the myopic heuristic performed better than competing heuristics in this case also. Results in the case of problems for which optimum could not be found are shown in Tables 4 and 5. Table 4 compares the mean deviation of normalized values of various heuristics from the best lower bound. Here again, myopic heuristic performs better than competing heuristics. Table 5 compares the mean value of myopic heuristic to the best available lower bound and best available upper bound. It is clear from this table that the myopic heuristic provided the best possible results among all heuristics tested.

In case of problems for which optimum found, it appears that the mean performance measure is at its worst for problems with tardiness factor 0.6 (Tables 2 and 3). This conclusion agrees with Srinivasan's conclusion [16] that problems with tardiness factor 0.65 were most difficult to solve. His conclusion was based on the computational time required to find optimum for the problems.

8. Conclusion

It is clear from our computational study that the new myopic heuristic developed by us is much better than any other heuristic tested. The heuristic is simple and easy to implement in most real life situations. The myopic heuristic can be used as a dispatching rule as well. In such a case, we merely determine which job is to be loaded on the machine next and make subsequent decisions as and when the machine becomes available for further loading. It is further possible to improve upon the schedule generated by the heuristic by checking for the local optimality among adjacent jobs. It is easy to build a procedure where we start with an initial schedule generated by our heuristic and make changes among adjacent jobs until no further improvement in the solution takes place. We are currently extending the application of our myopic heuristic to situations where we have more than one processor(identical processors in parallel). Further extensions in the area of generalized flow shops are being explored.

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. Baker, K.R.. *Introduction to sequencing and scheduling*. John Wiley & Sons, Inc., NewYork, 1974.

3. Baker, K.R. private communication. (telephonic conversation)

4. Baker, K.R. A Dynamic Priority Rule for scheduling againtst due dates. TIMS-ORSA Conference, Houston, October 12,1981.

5. Beshara, S.D., and M.J. Magazine. "Myopic Heuristics for Single Machine Scheduling Problems." *International Journal of Production Research 19*, 1 (Jan-Feb 1981), 85-95.

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

7. Elmaghraby, S.E.(ed). *Symposium on the Theory of Scheduling and Its Applications.* Springer-Verlag, NewYork, 1973.

8. Fisher, M.L. "A dual algorithm for the one machine sequencing problem." *Mathematical Programming 11* (1976), 229-251.

9. Lenstra, J.K. *Sequencing by Enumerative Methods.* Mathematisch Centrum, Amsterdam, 1977.

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

11. Picard, J.C. and Queyranne, M. "The time dependent travelling salesman problem and its applications to the tardiness problem in one machine scheduling." *Operations Research 26*, 1 (January-February 1978), 86-110.

12. RinnooyKan, A.H.G., Lagweg, B.J., and Lenstra, J.K. "Minimizing total costs in one machine scheduling." *Operations Research 23*, 5 (September-October 1975), 908-927.

13. Schild, A. and Fredman, I.J. "On Scheduling tasks with associated linear loss functions." *Managment Science 7* (1961), 280-285.

14. Schrage, L. and Baker, K.R. "Dynamic Programming solution of sequencing problems with precedence constraints." *Operations Research 26*, 3 (May-June 1978), 444-449.

15. Schweimer, J. "On the N-job, one machine, sequence-dependent scheduling problem with tardiness penalties: a branch and bound solution." *Management Science 18*, B (1972), 301-313.

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

APPENDIX

Consider the following relaxation of the single machine weighted tardiness problem: suppose that all jobs have unit processing times(if not, we split them into jobs of unit processing time and assign each the weight w_i/p_i. The due dates for these jobs are set at $d_i, d_i-1, d_i-2, \ldots\ldots d_i-p_i+1$). Let t_c be the completion time for the job J_i.

Consider the interchange of the current job J_i with another job J_j which is due to be completed at t_c+X. Since all jobs are of equal length, such interchange does not affect the completion time of any other job. Let w_i and d_i be the weight and the due date of job J_i.

PROPOSITION A.I: Let t_c be the completion time of J_i. Consider another job J_j completing X time units after J_i. Then, an optimal sequence should satisfy the following property–

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

PROOF: We have to consider eight subcases. These are as follows:

Case I: Both jobs are late in either position. Since both jobs are late in either position, the job with higher weight must precede the job with lower weight(Figure A.1)

It is clear that in this case the apparent priorities of both jobs are same as their weights and the condition is satisfied.

Case II: Both jobs are early in either position(Figure A.2). In this case, we are indifferent as to which job is scheduled first. Schedule first the job with highest apparent priority.

<u>Case III</u>: Both jobs are early in the current position and late in position t_c+X (Figure A.3).

Cost if J_i completes at t_c and J_j completes at t_c+X $= w_j(t_c+X-d_j) + 0$

$$= w_j X \left\{ 1 - \frac{(d_j - t_c)^+}{X} \right\}^+ \qquad \text{(I}$$

Cost if J_i completes at t_c+X and J_j completes at t_c $= w_i(t_c+X-d_i) + 0$

$$= w_i X \left\{ 1 - \frac{(d_i - t_c)^+}{X} \right\}^+ \qquad \text{(II}$$

Schedule J_i at t_c and J_j at t_c+X if

$$\text{(I)} \leq \text{(II)} \Rightarrow \quad w_i \left\{ 1 - \frac{(d_i - t_c)^+}{X} \right\}^+ \qquad\qquad w_j \left\{ 1 - \frac{(d_j - t_c)^+}{X} \right\}^+$$

<u>Case IV</u>: One job is late and the other is early in either position(Figure A.4). It is clear that the job that is late should be scheduled first. Note that the job that is early has zero apparent priority and the job that is late has full weight as its apparent priority.

<u>Cases V and VI</u>: One job is late in either position and the other is early in earlier position and late in later position(Figure A.5)

Cost if J_i completes at t_c and J_j completes at t_c+X $= w_j(t_c+X-d_j)$

Cost if J_j completes at t_c and J_i completes at t_c+X $= w_i(t_c+X-d_i) + w_j(t_c-d_j)$

We schedule J_i at t_c and J_j at t_c+X if

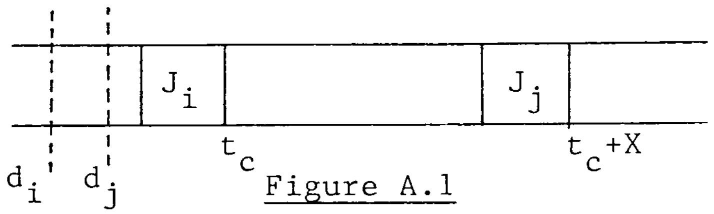

Figure A.1

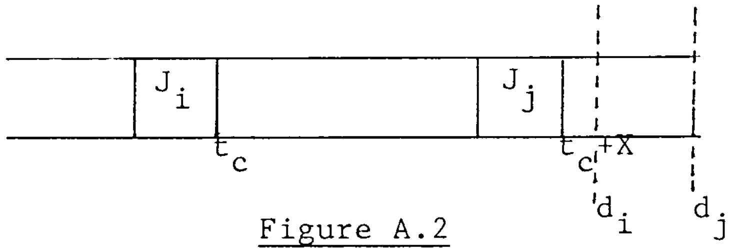

Figure A.2

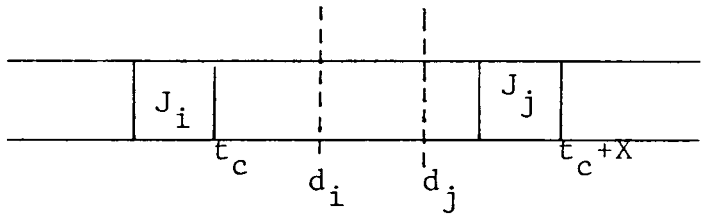

Figure A.3

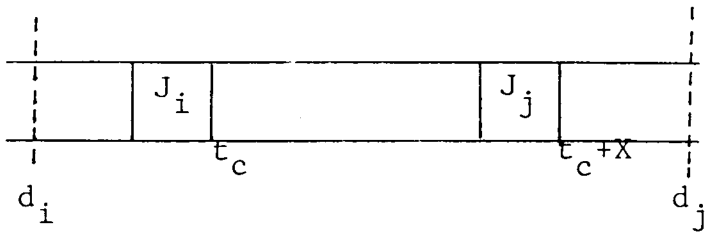

Figure A.4

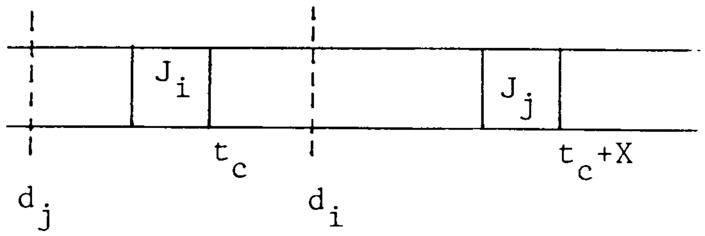

Figure A.5

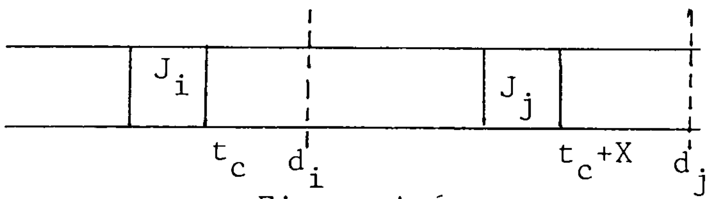

Figure A.6

$$w_j(t_c+X-d_j) \le w_i(t_c+X-d_i) + w_j(t_c-d_j)$$

$$w_j \le w_i \left\{ 1 - \frac{(d_i - t_c)}{X} \right\}$$

Since J_j is late at t_c and $d_i-t_c \le X$, the above expression may be rewritten as

$$w_i \left\{ 1 - \frac{(d_i - t_c)^+}{X} \right\}^+ \quad \ge \quad w_j \left\{ 1 - \frac{(d_j - t_c)^+}{X} \right\}^+$$

Cases VII and VIII: One job is early in either position and the other is early in earlier position and late in later position (Figure A.6). It is clear that J_i should be scheduled at t_c, since $d_j-t_c > X$, apparent priority of J_i will be greater than zero.

So, in all the cases discussed above, job with higher apparent priority should be scheduled in the current position.

PROPOSITION A.II: If all jobs have unit processng times and equal weights, the EDD sequence minimizes the average tardiness.

PROOF: Consider two adjacent jobs in an optimal sequence such that J_i precedes J_j and $d_i > d_j$.

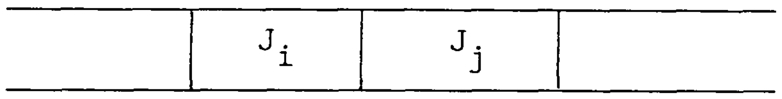

Figure A.7

Case I: Suppose both J_i and J_j are early or on time. Since J_j is early or on time and $d_i > d_j$, pairwise interchange does not degrade the solution.

Case II: Both J_i and J_j are tardy. Pairwise interchange does not degrade the solution since both processing times and weights are equal.

<u>Case III</u>: J_i is tardy and J_j is early or on time. This is impossible since $d_i > d_j$ and completion time of $J_i < J_j$.

<u>Case IV</u>: J_i is early or on time and J_j is tardy. If J_i is on time, then pairwise interchange does not degrade the solution. If J_i is early, then pairwise interchange improves the solution.

Thus, in all cases, pairwise interchange does not degrade the solution and, in fact, may improve it. Since our arguments employ only information about the individual jobs and not the location in the sequence[2] , the EDD sequence is optimal.