IIE Transactions (1997) **29**, 45–52

A heuristic to minimize the total weighted tardiness with sequence-dependent setups

YOUNG HOON LEE[1], KUMAR BHASKARAN[2], MICHAEL PINEDO[3]

[1]*Samsung Electronics Co. Ltd., Semiconductor Business, System Supports Group, P.O. Box 37 Suwon, Korea 449-900*
[2]*IBM T.J. Watson Research Center, P.O. Box 2018, Yorktown Heights, New York 10598, USA*
[3]*Department of Industrial Engineering and Operations Research, Columbia University, New York, New York 10027, USA*

Received April 1992, revised in August 1994 and accepted August 1995

We propose a three-phase heuristic for the problem of minimizing the total weighted tardiness on a single machine in the presence of sequence-dependent setup times. In the first phase a number of parameters characterizing the problem instance at hand are calculated. In the second phase we develop a schedule by using a new priority rule whose parameters are calculated based on the results of the first phase. Computational experiments show that this rule significantly outperforms the only other rule so far developed in the literature. The third phase consists of a local improvement procedure to improve the schedule obtained in the second phase. The procedure we suggest has been successfully implemented in an industrial scheduling system.

1. Introduction

Consider a single machine and n jobs that are all available for processing at time zero. Let the processing time, weight, and due date of job j be denoted by p_j, w_j, and d_j, respectively, where $j = 1,\ldots, n$. The completion time of job j depends on the schedule. If C_j denotes this completion time, then the tardiness T_j of job j is defined as $\max(C_j - d_j, 0)$. If job k immediately succeeds job j a setup time s_{jk} is incurred. Such setups are usually sequence dependent and s_{jk} need not be equal to s_{kj}. The objective is to find a sequence of jobs that minimizes the total weighted tardiness, i.e., $\sum w_j T_j$. If job j starts its processing at time zero, it is assumed to require a setup s_{0j}.

The case with sequence-independent setups has received considerable attention in the past, and has been shown to be strongly NP-hard (Lawler *et al.*, 1982). The problem is NP-hard in the ordinary sense when there are no setups and jobs have unit weights (Du and Leung, 1990); it then allows for a pseudo-polynomial time algorithm (Lawler, 1978). Potts and van Wassenhove (1982, 1985, 1987) developed efficient algorithms to solve the problem with no setups optimally for a limited number of jobs in a reasonable time. For an extensive survey of the total weighted tardiness problem with no setups, see Abdul-Razaq *et al.* (1990). It can be shown that the problem with arbitrary setup times and all n jobs having identical processing times is strongly NP-hard as well. Caroll (1965) developed a priority rule referred to as COVERT for the total tardiness problem. Vepsalainen and Morton (1987) and Ow and Morton (1989) developed an alternative rule, the Apparent Tardiness Cost (ATC) rule for the total weighted tardiness problem. This rule was applied to a single machine, to parallel machines and to various other machine environments. In their studies ATC compared quite favorably in performance with COVERT. COVERT and ATC schedule jobs one at a time, i.e., every time the machine becomes available, the job to be processed next is selected from the set of remaining available jobs. The decision is dynamic as the priority index of a job changes over time.

The ATC rule uses the following priority index at any instant t when the machine is available:

$$I_j(t) = \frac{w_j}{p_j} \exp\left[-\frac{\max(d_j - p_j - t, 0)}{k\,\overline{p}}\right].$$

Here $\overline{p}$ is the average processing time of all remaining jobs and k is a *look-ahead* parameter. The value of k is determined through experiments and for a single machine, according to Vepsalainen and Morton (1987), usually lies between 1 and 3. Observe that $\max(d_j - p_j - t, 0)$ is the slack of job j at time t. Every time the machine is available the indices of all remaining jobs are calculated and the job with the highest index is chosen to be processed next. If k approaches infinity the ATC rule reduces to the Weighted Shortest Processing Time first (WSPT) rule, which sequences the jobs in non-increasing order of w_j/p_j. If k approaches zero, and at most one job is overdue, it reduces to the Least Slack Remaining rule (LSR). If k approaches zero, and there is more than one job overdue, then the ATC rule reduces to

the WSPT rule among the overdue jobs. For intermediate values of k, the ATC rule constitutes a compromise between the WSPT rule and the LSR rule, i.e., the higher the w_j/p_j the more urgent the job, the larger the slack the less urgent the job.

The case with sequence-dependent setups has received very little attention in the literature so far. To our knowledge, there is only one reference with regard to this model in which problems are treated through a dispatching rule. Raman *et al.* (1989) suggested a modification of the ATC rule to take setup times into account. The priority index $I_j(t)$ is modified as

$$I_j(t,l) = \frac{w_j}{p_j + s_{lj}} \exp\left[-\frac{\max(d_j - p_j - s_{lj} - t, 0)}{k\,\overline{p}}\right],$$

where t again denotes the current time and l the index of the job just completed. The priority indices of all remaining jobs are computed and the job with the highest index is scheduled next. This rule modifies the ATC rule by replacing the processing time of a job with the sum of the processing time and the required setup time (which of course depends on the job just completed). We believe that this rule will work reasonably well in general. However, there may be certain situations where this rule will perform poorly. Consider the following situation: at time 0 there are two jobs remaining, jobs 1 and 2 with $p_1 = p_2 = 1$, $d_1 = d_2 = 3$ and $w_1 = 1$ and $w_2 = 3$. The job just completed at time 0 is job l and $s_{l1} = 0$, $s_{l2} = 2$, $s_{12} = 0$, $s_{21} = 5$. The rule would sequence the jobs in the order (2,1), whereas it is obviously better to sequence the jobs as (1,2). The reason that this rule results in a poor sequence in this situation is that the setup time appears in the numerator of the exponent. A large setup time decreases the slack of a job and increases its urgency, whereas it may be better to start a job with a very small setup time.

In this paper we suggest an alternative priority rule, the Apparent Tardiness Cost with Setups (ATCS) rule, for minimizing the total weighted tardiness when the jobs are subject to setup times. This rule is also a generalization of the ATC rule as it takes setup times into consideration. We show that the proposed rule performs better on average than that suggested by Raman *et al.* (1989), which will be referred to hereafter as the Raman's rule. This rule is a part of a three-phase heuristic procedure whose first phase consists of a statistical analysis of the problem instance at hand. The second phase constructs a schedule by using the ATCS rule, which has two parameters whose values depend on the results of the first phase. The third phase consists of a post-processing procedure designed to improve the solution obtained by the second phase. Our goal is to develop a procedure that works fast, say, scheduling 60 jobs within a minute on a PC, and is therefore applicable in practice.

This paper is organized as follows: in Section 2 a description of the overall heuristic is given. In Sections 3, 4, and 5 we describe three phases of the procedure. A summary, discussion of the results and some observations regarding future research can be found in Section 6.

2. The framework of the heuristic procedure

The priority index of the ATCS priority rule we suggest can be expressed as follows:

$$I_j(t,l) = \frac{w_i}{p_j} \exp\left[-\frac{\max(d_j - p_j - t, 0)}{k_1 \overline{p}}\right] \exp\left[-\frac{s_{lj}}{k_2\ \bar{s}}\right],$$

where t denotes the current time and l the index of the job just completed; $\bar{s}$ the average setup time; k_1 and k_2 are look-ahead or scaling parameters.

The ATCS rule, in contrast with Raman's rule, separates the effect of the remaining slack from the effect of the setup time. The priority of a job given by the WSPT ratio is exponentially discounted twice, once based on slack and again based on setup. These two effects are scaled separately by the parameters k_1 and k_2, which jointly provide the look-ahead capabilities of the ATCS rule. The values of the parameters k_1 and k_2 depend on the problem instance as they essentially perform scaling. In the first phase, the preprocessing procedure, we compute the values of a number of coefficients that characterize the instance at hand. By statistical analysis the relationship between the coefficients and the scaling parameters can be determined. In the second phase the values of the scaling parameters are determined. By using the ATCS rule with these parameter values the actual sequence of the jobs is generated. The third phase, the post-processing procedure, consists of the modification of the sequence to improve the quality of the schedule with respect to the objective function. The overall heuristic procedure is shown in Fig. 1.

3. Phase one: computation of coefficients

In the first phase three coefficients are computed in order to characterize the instance. The first coefficient is the Due Date Tightness factor τ, which is defined as

$$\tau = 1 - \bar{d}/C_{\max},$$

where $\overline{d}$ is the average of the due dates and $C_{\max}$ the makespan (the completion time of the last job to leave the system). It has been used before by Srinivasan (1971) and Baker and Martin (1974). It is clear that the makespan is schedule dependent because of the s_{jk} and therefore difficult to determine before the jobs are scheduled. To estimate τ it is necessary to have an estimate for $C_{\max}$. We will elaborate on the estimate for $C_{\max}$ later in this section.

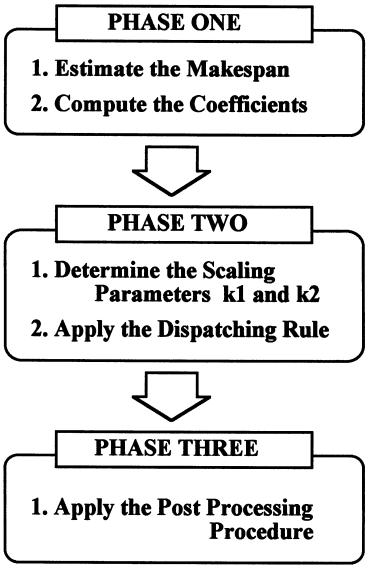

Figure 1 Framework of the heuristic procedure.

The second coefficient is the Due Date Range factor R, which is defined as

$$R = (d_{\max} - d_{\min})/C_{\max}.$$

It was introduced by Baker and Martin (1974) and used by Ow and Morton (1989). Again, we have to replace $C_{\max}$ by its estimator. Although τ gives an indication of the average tightness of the due dates, R provides a measure of the variability of the due dates with regard to the total workload.

The third coefficient is the Setup Time Severity factor η, which is defined as

$$\eta = \bar{s}/\bar{p},$$

where $\bar{s}$ is the average setup time.

As mentioned above, the makespan depends on the schedule owing to the fact that setup times are determined by the sequence. Therefore we have information about the makespan, $C_{\max}$, only after we determine a specific sequence. The makespan consists of two parts: the processing time of the jobs and the setup times between the jobs. Because every job has to be processed once, $n\bar{p}$ is the processing time component of $C_{\max}$. The setup time portion of $C_{\max}$ is, however, usually much smaller than $n\bar{s}$ because during the generation of the schedule a job with a smaller setup time is more likely to be selected.

We estimate $C_{\max}$ as $n(\bar{p} + \beta\bar{s})$, with $\beta \leq 1$. It is easy to see that the number of jobs, n, affects the value of β. From the $(n + 1) \times n$ matrix of setup time data, only n entries are selected, most of which are likely to have small values. Therefore one would expect the value of β to decrease as n increases (see Fig. 2). Another factor affecting β is the variability in the setup times. Obviously, if all setup times are the same, then β should be 1. As a measure of setup time variability we define the coefficient of variation c_v as

$$c_v = Var(s)/\bar{s}^2,$$

where Var(s) is the variance of the setup time data.

Through some experiments we attempt to develop an estimate of β as a function of n and c_v. We consider c_v values within the range [0, 1/3], because in our main experiments setup times are uniformly distributed (see Section 4). To have appropriate parameter values for k_1 and k_2 we start out with β equal to 0.5 and search for reasonable values for k_1 and k_2 (in Section 4 it is discussed how these values are determined). With these k_1 and k_2 values we do another set of experiments with same data sets, to find a good β value, and so on. It turns out that the makespan is not very sensitive to the values of k_1 and k_2 (the total weighted tardiness is, as shown in Section 4, very sensitive to the values of k_1 and k_2). Through these experiments we obtain the results shown in Fig. 2, which shows that β decreases in c_v and n. It is clear from the figure that 0.3 is a good estimate for β when $n = 60$ and $c_v = 1/3$.

In what follows we replace $C_{\max}$ by the estimator $n(\bar{p} + \beta\bar{s})$. The modified definitions for the Due Date Tightness factor and Due Date Range factor are

$$\tau = 1 - \bar{d}/n(p + \beta\bar{s})$$

and

$$R = (d_{\max} - d_{\min})/n(\bar{p} + \beta s).$$

In summary, the first phase computes the coefficient of variation c_v for the setup time data of the instance at hand and estimates the makespan of the schedule. Then it computes the values of the characterization coefficients, τ, R, and η.

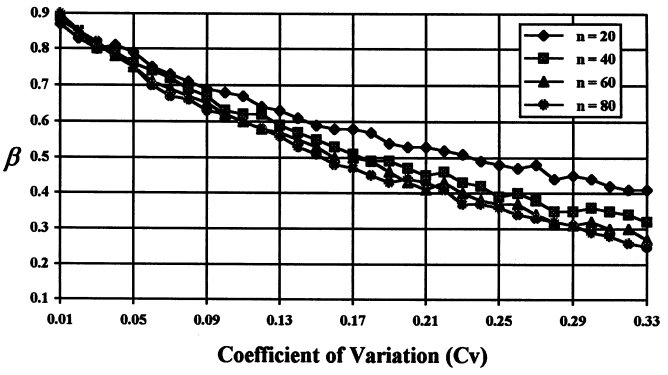

Figure 2 Values of β as a function of c_v and n.

4. Phase two: construction of the schedule

The parameters of the ATCS rule, k_1 and k_2, depend on the problem instance because they essentially perform scaling. It is advantageous to select proper values for k_1 and k_2 as a function of the values of τ, R, and η defined in the previous section. This dependency is not easy to determine. However, it is intuitive that k_2 is decreasing in η, because when η approaches 0 the setup times should not play a major role. We wish to find functions f_1 and f_2 which give appropriate values for k_1 and k_2 given τ, R, and η, i.e.,

$$k_1 = f_1(\tau, R, \eta)$$

and

$$k_2 = f_2(\tau, R, \eta).$$

An experimental study is conducted to establish these relationships.

4.1. *Experimental design*

The three problem coefficients (τ,R,η) and the two parameters of the ATCS rule constitute the dimensions of the experiment. The experiment covers 112 problem types consisting of seven τ values (τ = 0.3, 0.4,. . ., 0.9), four R values (R = 0.25, 0.5, 0.75, 1.0) and four η values (η = 0.25, 0.5, 0.75, 1.0). A problem type is uniquely determined by the three coefficients and the number of jobs n. In our study we set n to 60. The appropriate values of the coefficients are also somewhat affected by the number of jobs, but it appears from experiments that this influence is not as critical as that of the three coefficients. Within each problem type three problem instances are generated by using different random number seeds, resulting in a total of 336 instances. The processing times are assumed to be uniformly distributed over the interval [50, 150]. The mean processing time $\bar{p}$ is therefore 100. The mean setup time $\bar{s}$ is then determined by η and the setup times are uniformly distributed over the interval [0, $2\bar{s}$]. The due dates are generated from a composite uniform distribution based on R and τ. With probability τ the due date is uniformly distributed over the interval [$\bar{d} - R\bar{d}$, $\bar{d}$] and with probability $(1 - \tau)$ over the interval [$\bar{d}$, $\bar{d}$ + $(C_{\max} - \bar{d})R$]. This composite uniform has mean $\bar{d}$ and range R. The weight w_j of job j is generated from a uniform distribution on [0,10].

For each of the 336 instances generated, the ATCS rule is applied repeatedly over a (k_1, k_2) grid consisting of 512 points, i.e., ATCS is invoked 512 times for each of the 336 instances. The grid is obtained by setting k_1 = (0.2, 0.4,. . ., 6.4) and k_2 = (0.1, 0.2,. . ., 1.6). The values of the total weighted tardiness are then projected onto this grid. All k_1 values with which the minimum value of the objective function (say Z) were obtained are identified. The average of these k_1 values is then determined. The same is done with the k_2. These average values are referred to as $\overline{k_1}$ and $\overline{k_2}$, respectively. A range of the k_1 is then determined by considering all combinations (k_1, $\overline{k_2}$) that resulted in values of the objective function less than $(1+\alpha)Z$, where α is a number between 0 and 0.065 and a decreasing function of τ. The value 0.065 was considered appropriate after a thorough visual analysis of the experimental results. A range of k_2 is determined in the same way. Thus for each of 336 instances the best ranges of k_1 and k_2 are determined from the grid of 512 (k_1,k_2) combinations applied to the instance. So for each combination of τ, R, and η, there are three sets of best ranges of k_1 and k_2. For each combination of τ, R, and η, the union of the three ranges is then taken and the center points of the resulting intervals are determined. These center points are chosen as the recommended values for k_1 and k_2. These experiments lead to the results reported next.

4.2. *Results of the experiments*

The results presented here concern the general behavior of k_1 and k_2. These are then used to propose simple functions for selecting k_1 and k_2.

The behavior of k_1 is studied as a function of τ, R, and η. From the experiments it seems that k_1 is most sensitive to R among the three coefficients. It is also somewhat sensitive to τ and is hardly sensitive to η at all. In qualitative terms these dependencies can be summarized as follows (see also Fig. 3):

- for a given τ, k_1 displays a concave behavior in R with the peak at about R = 0.5;
- The value of k_1 tends to increase in τ up to approximately 0.6 and then decrease somewhat;
- for a given τ, k_1 increases slightly in η.

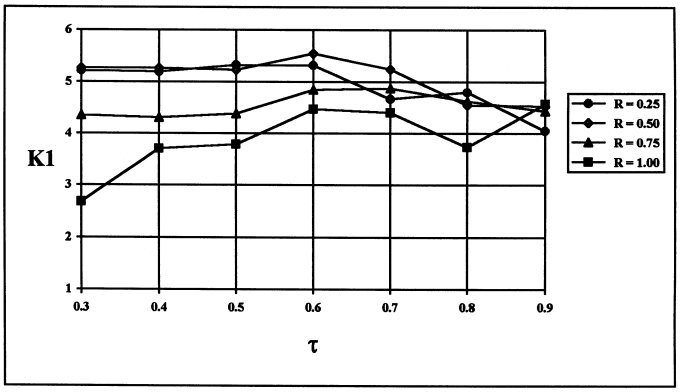

Figure 3 Plot of k_1 against τ and R.

Table 1. Percentage improvement obtained by ATCS with respect to Raman's rule, for $n = 60$

η	τ	R			
		0.25	*0.50*	*0.75*	*1.00*
0.25	0.3	40.8	53.7	55.5	65.3
	0.5	26.4	39.1	39.7	31.7
	0.7	12.5	13.4	14.7	13.5
	0.9	4.0	5.9	5.9	3.8
0.50	0.3	48.7	72.5	81.0	63.3
	0.5	32.5	40.2	62.6	61.7
	0.7	15.3	20.2	11.7	14.8
	0.9	3.6	3.4	6.8	5.6
0.75	0.3	69.8	78.5	88.7	90.4
	0.5	37.4	62.5	59.7	62.0
	0.7	14.4	20.9	16.6	22.5
	0.9	4.8	5.2	8.9	7.1
1.00	0.3	74.7	84.6	89.1	85.1
	0.5	36.1	63.9	68.1	57.3
	0.7	15.4	27.2	33.6	22.0
	0.9	5.9	5.9	8.1	6.5

The following rules could be used for the selection of a proper value of k_1:

$k_1 = 4.5 + R$ for $R \leq 0.5$;

$k_1 = 6.0 - 2R$ for $R \geq 0.5$.

In these rules the impacts of τ and η are not taken into account. Clearly, such rules may be modified or refined in the future to take the results of additional experiments into account.

The behavior of k_2 is also studied as a function of τ, R, and η. From our experiments it seems that k_2 is sensitive to τ and η and is invariant in R. Fig. 4 plots k_2 with respect to η for constant τ, ignoring the effects of R. The following observations can be made:

- for a given τ, k_2 displays a convex-decreasing behavior with respect to η;

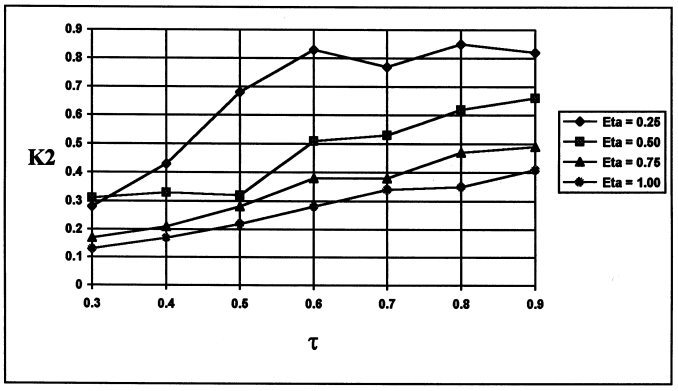

Figure 4 Plot of k_2 against τ and η.

Table 2. Average percentage improvement over Raman's rule obtained by ATCS for τ, R, η, and n

		n = 20	*n = 40*	*n = 60*	*n = 80*
$\tau =$	0.3	33.2	62.4	73.2	77.0
	0.5	18.7	39.6	47.8	50.7
	0.7	10.3	16.1	18.5	18.9
	0.9	4.9	5.6	5.6	5.9
$R =$	0.25	17.5	25.4	29.1	29.5
	0.50	17.7	32.1	36.3	38.0
	0.75	17.7	35.3	40.6	43.3
	1.00	14.2	31.1	39.0	41.7
$\eta =$	0.25	8.0	19.9	25.7	29.3
	0.50	14.7	30.3	35.8	36.6
	0.75	20.1	34.0	39.9	41.5
	1.00	24.3	39.5	43.7	45.0
Average		16.8	30.9	36.3	38.1

- for a given η, k_2 tends to increase with τ almost linearly.

A simple function that takes into account the effect of the τ and η on k_2 is the following:

$$k_2 = \tau/2\sqrt{\eta}.$$

The function for k_2 results in a very good fit. The behavior of k_1 is more complicated. It seems to be hard to capture all the trends with a relatively simple formula.

4.3. *Performance evaluation*

To see how good the schedules are that the ATCS rule generates we compare the performance of the ATCS rule with Raman's rule. Both rules are applied to a series of test problems. We pursued the following approach in order to perform a fair comparison (Raman *et al.* (1989) do not give any indication of how to choose their k value): first we conducted an experiment to find a good estimator for k as a function of τ, R, and η in Raman's rule. We performed similar experiments to those that we did for the ATCS rule. It seems that k is almost equally sensitive to the coefficients τ, R, and η (the fact that k depends on all three coefficients is clear because there is only one scaling parameter in Raman's rule). In qualitative terms these relationships can be summarized as follows:

- k is increasing in η with a slope of about 1;
- k is decreasing in τ and R with a slope of about 1.

The following rule could be used for the selection of a proper value of k:

$$k = 5.5 - \tau - R + \eta.$$

Table 3. Comparison of the performances of two rules

		n = 20	*n = 40*	*n = 60*	*n = 80*
ATCS better	Number of cases	1517	1861	1912	1917
	Best practice	100%	100%	100%	100%
Raman's better	Number of cases	403	59	8	3
	Best practice	99.52%	60.57%	3.81%	2.57%

Notes:
1. Number of cases: among 1920 cases (30 cases for each value of τ, R, η and n).
2. The value of the best practice shows the relative improvement over the alternative schedule. 100% means that the better schedule is optimal with the total weighted tardiness zero.

Table 4. Percentage performance improvement of post-ATCS procedure

		Swap			*Insertion*		
		LST	*LWT*	*RND*	*LST*	*LWT*	*RND*
CPU (seconds)		0.125	0.130	0.133	0.129	0.133	0.128
τ =	0.3	8.26	21.48	3.87	6.63	21.06	2.00
	0.5	0.43	1.24	0.97	0.25	2.45	0.61
	0.7	0.06	0.04	0.29	0.07	0.12	0.20
	0.9	0.01	0.01	0.01	0.01	0.04	0.01
R =	0.25	0.86	3.51	0.45	0.65	3.88	0.28
	0.50	1.80	3.44	1.05	0.33	4.35	0.66
	0.75	4.29	8.86	2.03	3.23	7.70	1.29
	1.00	0.82	6.96	0.60	1.75	7.72	0.59
τ =	0.25	2.13	4.61	1.16	1.40	4.30	0.45
	0.50	2.20	6.68	1.85	2.26	6.07	1.24
	0.75	1.96	5.03	0.80	1.44	6.16	0.51
	1.00	2.48	6.44	1.32	1.86	7.12	0.62
Average		2.19	5.69	1.29	1.74	5.91	0.70

With this function for the scaling parameter the performance of Raman's rule is compared with the performance of the ATCS rule. Instances are generated for τ = (0.3, 0.5, 0.7, 0.9), R = (0.25, 0.5, 0.75, 1.0) and η = (0.25, 0.5, 0.75, 1.0). Within each problem type five instances are generated by using different random number seeds to get the average improvement of the ATCS rule over Raman's. Table 1 shows the average improvement obtained by the ATCS rule over Raman's for various cases of τ, R and η.

It is observed that considerable improvement can be achieved for low values of τ: the due dates are relatively loose, and hence the total weighted tardiness is small. In other words, the total weighted tardiness as an objective function is less significant for low τ. For larger τ values it is harder to get a significant improvement. The improvement obtained for each value of R and η is almost the same. We also compared the performance of two rules for various number of jobs, say, n = 20, 40, 60, and 80. Table 2 summarizes the average improvement for each value of τ, R, η, and n. Notice that the formulae developed for n = 60 are used for n = 20, 40 and 80 in our experiment. Table 3 shows that the ATCS rule is better in most cases.

5. Phase three: the post-processing procedure

In phases one and two, we determine a sequence for any given instance. In the third phase we attempt to modify this sequence to obtain a better schedule. Modifications can be made in a number of different ways. We describe here some procedures that lead to improvement after a few iterations. Basically, a sequence may be modified in two ways, through an insertion or through a swap.

1. *Insertion procedure*: Two jobs A and B are selected. Job A is inserted immediately after job B if it improves the objective function. As a candidate for job A, it seems to be advantageous to select either the job with the largest contribution to the total setup time or the job with the largest contribution to the total weighted tardiness in the ATCS sequence. We may also choose job A randomly and compare the performance of such a selec-

tion procedure with other methods. These three selection methods are called LST (Longest Setup Time), LWT (Largest Weighted Tardiness) and RND (Random), respectively in what follows. A candidate for job B is selected from among 20 jobs nearest to job A. The value of the objective function of the sequence, which is constructed by inserting job A immediately after the candidate of job B, is computed and compared with each one of 20 cases. Job B is selected such that the resulting sequence has the minimum value of the objective function among 20 cases. The computation and comparison can easily be done sequentially. We restrict the set of jobs from which job B is chosen because it seldom occurs that the best job B candidate is far away from the job A selected.

2. *Swap procedure:* We can also modify the sequence by swapping jobs. Two jobs A and B are selected and interchanged while all other jobs keep the same sequence. We select job A in the same way as in the insertion procedure, i.e., the job with the longest setup time (LST), with the largest penalty (LWT) or randomly (RND). From the 20 jobs nearest to the first job, the second job is selected by computing the objective functions of the sequence obtained through a swap, which is similar to the insertion procedure. A method is characterized only by the way the first job is selected, not by the way the second job is selected.

We experimented with the six methods described above for τ = (0.3, 0.5, 0.7, 0.9), R = (0.25, 0.5, 0.75, 1.0) and η = (0.25, 0.5, 0.75, 1.0). For any given τ, R, and η, we generated five instances, computed the average improvement obtained with each method and made a comparison. We also took the CPU time (on a Sun Sparc-IPC workstation) needed to run one instance for each method into account. We performed the improvement procedure, insertion or swap, three times for each method. The CPU times taken for every method are shown in Table 4. The time taken to apply the post-processing procedure is approximately 0.125 CPU seconds for each of the three methods, whereas the main ATCS procedure requires 0.291 CPU seconds. Notice that finding the job with the longest setup time or the largest penalty does not take more time than selecting a job randomly; this can be done by storing the data of the setup time or the penalty while generating a sequence in Phase Two. For each of the three methods and each combination of the three coefficients we found the improvement as a percentage of the value of the total weighted tardiness of the ATCS sequence. In Table 4 each entry indicates the normalized improvement of each method, which can be defined as the improvement divided by the standard CPU time (0.125 CPU seconds).

As we can see from Table 4, in general the LWT method performs better than the LST method in both the swap and the insertion procedure, whereas the method using randomly selected jobs performs poorly. There is no significant difference between the swap and the insertion method, and considerable improvement can be achieved for low values of τ. It is hard to obtain a good schedule in a small amount of computer time when τ is high. The improvement obtained is somewhat increasing in R, but the value of R at which the largest improvement is achieved is not always 1. From the experiments it appears that it is useful to apply the post-processing procedure when τ is low and R relatively high.

6. Summary and discussion

In this paper we suggest a priority rule for the total weighted tardiness problem with sequence-dependent setup times. The priority index is defined with parameters that are functions of coefficients characterizing the instance. The sequence is generated in three phases. Phase one, the preprocessing procedure, consists of the computation of the three coefficients, the due date tardiness factor, the due date range factor and the setup time severity factor. With these values, in Phase two the dispatching rule is applied to generate the sequence. The performance of the dispatching rule is compared with Raman's rule, which is the only rule so far in the literature for this problem. Our rule is on the average better by more than 30% with respect to the value of the objective function when the number of jobs is large (more than 40). Phase three, the post-processing procedure, refines the sequence obtained in the second phase. We suggest six methods that lead to significant improvement in relatively short computation time.

One can use the heuristics presented in this paper in various ways. First, one can consider implementations where at time zero one computes values of τ, R, and η and determines the values of k_1 and k_2 once for all. However, if the number of jobs is very large one can imagine that after scheduling a (maybe small) number of jobs the characteristics of the remaining jobs have changed. It might be an idea then, taking the current time into account, to recompute τ, R, and η, redetermine k_1 and k_2 and proceed with these new numbers. Such an approach could be desirable if there is high variability in the data. After processing a very unusual job (unusual in the sense of its data) the average of the remaining jobs may change significantly.

In practice the heuristics presented in this paper can of course be used in conjunction with other optimization techniques. For example, it may be used in conjunction with enumeration : the heuristic determines the n jobs of the highest priority; a complete enumeration is then done to determine in what order these n jobs should be processed. The heuristics presented in this paper can also be used in conjunction with techniques such as simulated annealing (Matsuo *et al.*, 1989; Lee *et al.*, 1995) or Tabu search (Barnes and Chambers, 1991). The

results of the heuristic can be used as an initial solution for a simulated annealing or Tabu search routine. Notice that a significant improvement can be obtained in the third phase with a simple swap/insertion procedure. If a simulated annealing algorithm or Tabu search that requires more computation time is applied, a bigger improvement could be achieved.

When jobs have different release dates the heuristics developed in this paper can still be used. Every time the machine is free one selects the job available for processing with the highest priority index. The τ, R, and η are computed taking into account all jobs, that is, including those still to be released.

The ATCS procedure described in this paper has been implemented in a scheduling system, the so-called BPSS system, which is currently in operation in a number of factories in the USA (Adler *et al.*, 1993). This system schedules jobs in a complicated machine environment by using an approach consisting of multiple phases. In one of these phases the ATCS rule is used to solve a scheduling problem with a number of machines in parallel.

References

Abdul-Razaq, T.S., Potts, C.N. and van Wassenhove, L.N. (1990) A survey of algorithms for the single machine total weighted tardiness scheduling problems. *Discrete Applied Mathematics* **26**, 235–253.

Adler, L., Fraiman, N.M., Kobacker, E., Pinedo, M., Plotnitcoff, J.C. and Wu, T.P. (1993) BPSS: a scheduling system for the packaging industry. *Operations Research*, **41**, 641–648.

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

Barnes, J.W. and Chambers, J.B. (1991) Solving the jobshop scheduling problem using tabu search. Technical Report ORP91-06, University of Texas, Austin.

Caroll, C.D. (1965) Heuristic sequencing of jobs with single and multiple components. Ph.D. Thesis, MIT, Cambridge, MA.

Du, J. and Leung, J.Y. (1990) Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*, **15**, 483–494.

Lawler, E.L. (1978) A pseudo-polynomial time algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, **1**, 331–342.

Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G. (1982) ''Deterministic and Stochastic Scheduling''. Recent developments in deterministic sequencing and scheduling: a survey, deterministic and stochastic scheduling. Dempster, M.A.H., Lenstra, J.K. and Rinnooy Kan, A.H.G. (eds.), Reidel, Dordrecht, 35–73.

Lee, Y.H. and Pinedo, M. (1995) Scheduling jobs on parallel machines with sequence-dependent setup times. To appear in *European Journal of Operational Research.*

Matsuo, H., Suh, C.J. and Sullivan, R.S. (1989) A controlled search simulated annealing method for the single machine weighted tardiness problem. *Annals of Operations Research*, **21**, 85–108.

Ow, P.S. and Morton, T.E. (1989) The single machine early/tardy problems. *Management Science*, **35**, 177–191.

Potts, C.N. and van Wassenhove, L.N. (1982) A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters*, **1**, 177–182.

Potts, C.N. and van Wassenhove, L.N. (1985) A branch and bound algorithm for the total weighted tardiness problem. *Operations Research*, **33**, 363–377.

Potts, C.N. and van Wassenhove, L.N. (1987) Dynamic programming and decomposition approaches for the single machine total tardiness problem. *European Journal of Operational Research*, **32**, 405–414.

Raman, N., Rachamadugu, R.V. and Talbot, F.B. (1989) Real time scheduling of an automated manufacturing center. *European Journal of Operational Research*, **40**, 222–242.

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

Vepsalainen, A. and Morton, T.E. (1987) Priority rules for jobshops with weighted tardiness costs. *Management Science*, **33**, 1035-1047.

Biographies

Young Hoon Lee received the B.S. degree in industrial engineering from Seoul National University, Korea, and the M.Sc. and Ph.D. degrees in industrial engineering from Columbia University, New York. He is a senior manager in charge of the System Support Group of Samsung Electronics Co., Semiconductor Business. His research interests are in production planning and control, production scheduling, with emphasis on applications and system developments in the semiconductor industry. His recent articles have appeared or are going to appear in *IIE Transactions, European Journal of Operational Research*, and *Production Planning and Control.*

Kumar Bhaskaran has been active in research and consulting in the areas of production scheduling, inventory management, and decision support systems for supply chains for the past eight years. He has a Ph.D. in Engineering Science from the Rensselaer Polytechnic Institute. He is an active member of IEEE and INFORMS. Currently, he is a member of the research staff in IBM T.J. Watson Research Center, Yorktown Heights, New York.

Michael Pinedo received the Ir. degree in mechanical engineering from the Delft University of Technology, the Netherlands in 1973, M.Sc. and Ph.D. degrees in industrial engineering from the University of California at Berkeley in 1978. He is Professor in Industrial Engineering and Operations Research at Columbia University. His research interests are in production systems modeling, queuing theory, scheduling theory and scheduling systems development. He has written or jointly written numerous technical papers on these topics and is the author of the book *Scheduling: Theory, Algorithms and Systems*. Over the last decade he has been very involved in industrial systems development. He supervised the design, development and implementation of two scheduling systems for the International Paper Company. He also actively participated in the development of scheduling systems at Philips Electronics, Siemens, and at Merck. He is a departmental editor for *IIE Scheduling and Logistics* (covering scheduling) and an associate editor of *Naval Research Logistics*. He has been an area editor of *Operations Research* (covering stochastic processes).