ALGORITHMS FOR SINGLE MACHINE SCHEDULING PROBLEMS

MINIMIZING TARDINESS AND EARLINESS

Candace A. Yano
Yeong-Dae Kim

Department of Industrial & Operations Engineering
The University of Michigan
Ann Arbor, Michigan 48109-2117

Technical Report 86-40

October 1986

ALGORITHMS FOR SINGLE MACHINE SCHEDULING PROBLEMS

MINIMIZING TARDINESS AND EARLINESS

ABSTRACT

Most scheduling research has been done with one criterion, but in some situations two or more criteria should be considered at the same time. In this paper, single machine scheduling problems with the objective of minimizing the sum of tardiness and earliness are considered. An algorithm for the optimal completion times of jobs is developed, and some properties of optimal sequences are discussed. These properties are used to develop both optimal and heuristic procedures. Computational results for problems with up to 40 jobs are reported.

ALGORITHMS FOR SINGLE MACHINE SCHEDULING PROBLEMS MINIMIZING TARDINESS AND EARLINESS

In many practical scheduling problems, costs arising from both earliness and tardiness of the individual jobs must be minimized. For example, in production systems in which there are shipping dates for orders, inventory carrying costs are incurred if jobs are finished earlier than the shipping dates, and shortage costs (penalty costs or backorder costs) are incurred if jobs are finished later than the shipping dates. To minimize total costs, tardiness and earliness of jobs should be minimized. In this paper earliness and tardiness are defined as $\max(d_i - C_i, 0)$ and $\max(C_i - d_i, 0)$ respectively, where d_i is the due date and C_i is the completion time of job i.

There has been some research for the single machine scheduling problems with multiple performance criteria related to inventory carrying costs and shortage costs. Emmons(1975a) develops a branch-and-bound algorithm for the bicriterion scheduling problem in which minimizing the number of tardy jobs is the primary objective and minimizing mean flow time is the secondary objective. Emmons(1975b) also gives some properties of solutions for the problems with the dual criteria of mean flow time and maximum penalty, where the penalty reflects the cost of completing a job at a certain time. Later, Van Wassenhove and Gelders(1980) consider the bicriterion problem to minimize holding cost and maximum tardiness. The set of efficient points is characterized and a pseudo-polynomial algorithm to enumerate all these points is given. These objectives are combined as a single objective which is a convex combination of the two in Sen and Gupta(1983). Algorithms to generate efficient points are developed for multicriterion performance measures in other research such as Van Wassenhove and Baker(1982) for time/cost trade-offs, Lin(1983) for mean tardiness and mean flow time, and Nelson et al.(1986) for mean flow time, maximum tardiness, and number of tardy jobs.

Very few papers have been published which consider tardiness and earliness.

Sidney(1977) presents an optimal algorithm where the objective is to minimize the maximum of the costs of earliness and those of tardiness, where the costs are nondecreasing functions of earliness and tardiness, respectively. Following this, a more efficient algorithm was developed for the same problem by Lakshminarayan et al.(1978). In these two papers earliness is defined as the difference between the target start time and the actual start time. Townsend(1978) develops a branch-and-bound procedure for the objective of minimizing total penalty, where the penalty of each job is expressed as a quadratic function of completion time. Since tardiness plus earliness is a piecewise linear convex function of completion time, his problem is similar to ours in some sense. Using a similar bounding technique Gupta and Sen(1983) presents a branch-and-bound and a heuristic algorithms for minimizing a quadratic function of job lateness. Here they do not allow machines to be idle. Differently from the others, Panwalkar et al.(1982) consider the problem in which the due dates are decision variables. They develop a polynomial algorithm to find the due dates and the sequence to minimize the costs resulting from due dates, earliness, and tardiness.

In this paper two problems are considered, which are $n/1//\bar{T}+\bar{E}$ and $n/1//\Sigma(\tau_i T_i + \epsilon_i E_i)$. Here n/m/A/B indicates the scheduling problem in which n, m are the numbers of jobs and machines respectively, A is the flow discipline, and B is the performance criterion. T_i and E_i refer to the tardiness and the earliness of job i, and τ_i and ϵ_i are their weights. Therefore these problems are single machine scheduling problems with objectives of minimizing mean tardiness plus earliness ($\tau_i = \epsilon_i = 1$ for all i), and minimizing weighted sum of tardiness plus earliness where the weights of tardiness and earliness of each job are different, respectively. The first problem is a special case of the second. Note that the mean tardiness problem is a subset of the second problem but it is unrelated to the first. The relationships among these and other related problems are shown in Figure 1. A special case of the $n/1//\bar{T}+\bar{E}$ problem, which is $n/1//\bar{T}+\bar{E}$ with equal due dates, has been discussed in Kanet(1981), Sundararaghavan and Ahmed(1984), and

Bagchi et al.(1986). Even though this problem may not occur frequently in practice, it may be important from a theoretical perspective. If this problem is NP-complete, all the problems mentioned in this paper are NP-complete since this problem is a special case of the others.

Section 1 presents an algorithm for determining completion times of jobs for a given sequence. In Section 2, some properties of optimal job sequences for the $n/1//\bar{T}+\bar{E}$ problem are discussed and a branch and bound algorithm and a heuristic algorithm are presented. Section 3 discusses the general problem, i.e., $n/1//\Sigma(\tau_i T_i + \epsilon_i E_i)$, and computational results are presented in Section 4. Finally some concluding remarks appear in Section 5.

1. DETERMINING OPTIMAL COMPLETION TIME FOR EACH JOB IN A SEQUENCE

When a sequence is given, an optimal schedule can be obtained easily by using a linear program(LP), since the technical constraints(precedence relationships) are all given. An LP formulation is

$$\text{Min } \Sigma (E_i + T_i)$$

$$\text{s.t. } d_i - C_i = E_i - T_i , \quad \text{for all } i$$

$$C_{(i)} - C_{(i-1)} \geq p_{(i)} , \quad i=2 \text{ to } n$$

$$C_i, E_i, T_i \geq 0 , \quad \text{for all } i$$

where (i) is i-th job in the given sequence, $p_{(i)}$ is processing time for the i-th job, and the other notation is the same as before.

Earliness and tardiness of a job can be expressed by the difference between the completion time and the given due date for the job as in the first constraint. The second constraint assures that no more than one job can be processed on the machine at any time. Since a sequence is given, disjunctive constraints are not necessary. Note that there are 3n variables and 2n-1 constraints in this LP formulation. However, the best known algorithm for LPs requires pseudo-polynomial time.

A more efficient algorithm which is strongly polynomial is presented here. The following notation is used in this algorithm. Throughout the remainder of this section subscript i refers to the i-th job in the given sequence.

p_i : processing time for the i-th job in the sequence

d_i : due date of the i-th job in the sequence

C_i : completion time of i-th job in the sequence

S_i : slack time(machine idle time) between job i and job i+1

$$S_i = C_{i+1} - p_{i+1} - C_i$$

G_i : gradient of T+E (tardiness plus earliness) of job i when job i is moved 1 unit time later

$D_i = \max (d_i - C_i, 0)$

TG : gradient of T+E of a partial sequence when the job being considered is moved 1 time unit later at a stage of the algorithm

s : start time of the first job in partial schedule at any stage of the algorithm

u_i : $\sum_{k=1}^{i-1} p_i$, the earliest possible time job i can start

Note that TG for any partial sequence with no inserted idle time between jobs is a nondecreasing step function of the completion time, and therefore the sum of earliness and tardiness for the partial sequence is piecewise linear convex as long as there is idle time following the partial sequence.

Now the algorithm can be stated. In this algorithm, the jobs are considered in reverse order of the given sequence. Using G_i and TG, we shift jobs as late as possible without increasing T+E. This provides more time for the jobs which are to be placed prior to them.

Algorithm 1.

(0) Initialization

Let $C_n = d_n$, $s = C_n - p_n$, $G_n = 1$, $TG = G_n$, $S_n = \infty$.
Let $j = n - 1$, and go to (1).

(1) If $j = 0$, go to (4).

If $j > 0$, if $s \geq d_j$, go to (2),
if $s < d_j$, let $D_j = d_j - s$, $C_j = s$, $G_j = -1$,
$TG = TG + G_j$, and go to (3).

(2) Let $C_j = d_j$, $s = C_j - p_j$, $S_j = s - d_j$, $G_j = 1$.
Reset $TG = G_j$.
Let $j = j - 1$, and go to (1).

(3) If $TG > 0$, let $j = j - 1$, and go to (1).

If $TG \leq 0$, shift jobs to right until further shifting is not possible without making TG positive, and update values of C_i, D_i, s, G_i, TG if needed. Let $j = j - 1$, and go to (1).

(4) If $S_1 \geq 0$, stop. The resulting schedule is optimal. Otherwise, shift jobs to right to make $S_1 = 0$. Terminate. ■

When we visit step 2, there is a positive slack time between job j and job j+1. At this time we reset $TG = G_j$, since job j-1 would not be affected by the jobs after j. At step 3, if $TG > 0$, we cannot reduce T+E of the jobs scheduled so far by shifting jobs to the right or left. If $TG \leq 0$, shifting jobs to right does not increase (and may decrease) T+E of the jobs scheduled. This shift increases TG. When the amount of shift is equal to $S_i > 0$ for some i, the slack time between jobs i and i+1 becomes 0 and TG should be recalculated by adding the gradients of jobs in the partial sequence which includes job i and has no inserted slack time between jobs.

Figure 2 shows how this algorithm works in a simple example. The given sequence in this example is (1,2,3,4,5,6), and the problem data are shown as a figure at the top of Figure 2. The rectangles denote jobs and their widths denote the processing times of jobs.

The following theorem shows the efficiency and optimality of this algorithm.

Theorem 1. Algorithm 1 obtains an optimal schedule when a sequence is given, and the worst case computational complexity is $O(n^2)$.

Proof. Since the sequence is given, we can only shift jobs forward or backward without changing the order. Consider any subset of jobs with no inserted slack time between jobs in the solution resulting from Algorithm 1. Since $\Sigma\ (E_i + T_i)$ for the subset is piecewise linear convex, and in the solution the jobs in the subset have been shifted up to the point where TG becomes positive (from zero or negative), we cannot improve the solution by shifting the jobs in either direction. Therefore the solution is optimal.

If job j is considered and we visit step 3, $TG=0$. To find the amount of shift that makes TG positive, we only have to find the minimum value among S_{j+p} and D_i for $i=j, j+1,\cdots, j+p$, where jobs $j, j+1,\cdots, j+p$ are in the same subset described as above. This needs effort of O(p) where p is not greater than n-j. After this we need to update the values C_i , D_i , G_i , TG, which needs O(n-j) effort at most. Therefore every visit to step 3 needs O(n) effort. At step 4, if a shift is needed we can do it in O(n) as follows. Calculate u_i and $C_i - p_i$ and find $k = \min [\ i \mid u_i \leq C_i - p_i\ ]$. Reset $C_i = u_i + p_i$ for $i=1,2,\cdots,k-1$. This needs O(k) effort, and $k \leq n$. Steps 1 and 2 need O(1) effort for each visit.

Every step can be visited at most n times in the entire algorithm. Therefore the overall computational complexity of this algorithm is $O(n^2)$ in terms of basic arithmetical operations such as additions, comparisons, or look-ups. ■

We now have an algorithm to determine the optimal completion times of jobs for a given sequence. It can be used as a subroutine in determining good or optimal sequences of jobs. In the next section we develop properties of the optimal job sequences.

2. ALGORITHMS FOR $n/1//\bar{T}+\bar{E}$ PROBLEM

In this section, algorithms for the $n/1//\bar{T}+\bar{E}$ problem are presented. Since a polynomial algorithm has not been found for this problem, a branch-and-bound algorithm and a heuristic algorithm are developed. First some properties of the optimal sequence of jobs which are useful for these algorithms will be discussed.

Lemma 2. If there is a conflict between 2 and only 2 jobs when the jobs are placed as $C_i = d_i$ for both jobs, and if $p_i < p_j$, then the following statements are true.

(Case 1) If $d_i > d_j$, then job j should precede job i.

(Case 2) If $d_i < d_j$

(Subcase 2.1) If $p_i + (d_j - d_i) \geq p_j$, then i should precede j.

(Subcase 2.2) If $p_i + (d_j - d_i) < p_j$, let $A = p_j - p_i - (d_j - d_i)$.

If $A \leq d_j - d_i$, i should precede j, otherwise, j should precede i. (See Figure 3.)

Proof. Let $TE_{ij} = \mathrm{Min}_{i,j}\,[\,T_i + E_i + T_j + E_j\,]$, where i precedes j.

Using algorithm 1 we can get TE_{ij} and TE_{ji}.

(Case 1) $TE_{ij} = p_j - (d_i - d_j)$, $TE_{ji} = p_i - (d_i - d_j)$.

Since $p_i < p_j$, $TE_{ji} < TE_{ij}$.

(Case 2) $TE_{ij} = p_j - (d_j - d_i)$, $TE_{ji} = p_i + (d_j - d_i)$.

(Subcase 2.1) Since $p_i < p_j$ and $d_i < d_j$, $TE_{ji} \geq p_j \geq TE_{ij}$.

(Subcase 2.2) When $A \leq d_j - d_i$, $TE_{ij} = A + p_i \leq p_i + (d_j - d_i) = TE_{ji}$.

When $A > d_j - d_i$, $TE_{ij} = A + p_i \geq p_i + (d_j - d_i) = TE_{ji}$. ■

By considering all cases in the proof of Lemma 2, and by using Algorithm 1, the following corollaries can be proven easily. They are, therefore, stated without proof.

Corollary 3. If there is a conflict between 2 jobs when the jobs are placed as $C_i = d_i$ for both jobs, the sum of tardiness and earliness is not less than the time during which the two jobs overlap.

Corollary 4. In the same situation as in corollary 3, there may be alternate optimal schedules and there exists an optimal schedule such that either $C_i = d_i$ or $C_j = d_j$.

These properties are useful in getting bounds for subproblems in the B&B algorithm. The next property can be used for pruning some subproblems. This property shows some dominance rules of sequences by considering adjacent jobs in cases where both jobs should be finished earlier than their due dates, or where both jobs are started after their due dates because of succeeding or preceding jobs.

Lemma 5. In adjacent jobs i and j,

(1) If there is a constraint such that $t = \max(C_i, C_j) \le \min(d_i, d_j)$,

then i should precede j when $p_i > p_j$, and j should precede i when $p_i < p_j$.

(2) If there is a constraint such that $\min(s_i, s_j) \ge \max(d_i, d_j)$,

then i should precede j when $p_i < p_j$, and j should precede i when $p_i > p_j$.

(3) If there is a constraint such that $\min(s_i, s_j) + \min(p_i, p_j) \ge \max(d_i, d_j)$,

then i should precede j when $p_i < p_j$, and j should precede i when $p_i > p_j$,

where s_i is the start time of job i. (See Figure 4.)

Proof.

(1) $TE_{ji} = d_j - d_i + p_i + 2(d_i - t)$,

$TE_{ij} = d_j - d_i + 2(d_i - t) + p_j$,

$\therefore TE_{ij} - TE_{ji} = p_j - p_i$. Hence the results follow.

(2) $TE_{ij} = 2(t - d_j) + 2p_i + (d_j - d_i) + p_j$,

$TE_{ji} = 2(t - d_j) + 2p_j + (d_j - d_i) + p_i$,

$\therefore TE_{ij} - TE_{ji} = p_i - p_j$. Hence the results follow.

(3) $TE_{ij} = t + p_i - d_i + t + p_i + p_j - d_j$,

$TE_{ji} = t + p_j + p_i - d_i + t + p_j - d_j$,

$\therefore TE_{ij} - TE_{ji} = p_i - p_j$. Hence the results follow. ∎

Now a branch-and-bound algorithm using the above properties can be stated. Each node of the branching tree is associated with a partial sequence which will be placed at the end of the whole sequence. That is, a node at the p-th level of the tree corresponds to a partial sequence <p>,<p-1>,<p-2>,···,<1>, where <i> represents i-th job from the end, and one of the remaining n-p jobs is to be selected for <p+1>. Let J be the set of all jobs, σ be a partial sequence being placed at the end of the sequence, and PS be the set of jobs in σ .

Algorithm 2. (A branch-and-bound algorithm)

Branching

Select a node with the least lower bound in branching tree for branching.

Bounding

(1) Bound B_1 for jobs in PS

B_1 is the optimal solution (minimum value of $\Sigma(T_i+E_i)$) obtained from Algorithm 1 for the sequence σ under the constraint that the earliest possible start time of the first job in PS is $\sum_{i \in J \backslash PS} p_i$ instead of 0.

(2) Bound B_2 for job in J\PS

B_2 is the sum of the time intervals during which two or more jobs overlap, when we place all the jobs in J\PS to satisfy $C_i = d_i$. This bound can be justified by Corollary 3

Then the lower bound of the node associated with the partial sequence σ will be

$$B = B_1 + B_2 .$$

Pruning

If there is any adjacent pair of jobs i, j in the partial sequence σ, such that $\min[s_i, s_j] + \min[p_i, p_j] \geq \max[d_i, d_j]$, prune that partial sequence. This can be justified since there always exists a better sequence σ^* which is identical to σ except for the order of jobs i and j. ■

To improve the efficiency of the algorithm the lower bounds need to be sharper(larger). The following lemma and the argument following it help to improve the lower bound B_2.

Lemma 6. If there are conflicts among 2 or more jobs when a set of jobs is placed as $C_i = d_i$ for all i in the set, the sum of tardiness and earliness is not less than $\sum_{k \geq 2} (k-1)t_k$, where t_k is the length of time during which k jobs overlap.

Proof. It is obvious that the optimal objective function value for a set of jobs is not less than the sum of the objective value of the subsets of jobs which make the set itself. We can divide a set of jobs into several subsets which are separated at times when there are no overlapping jobs (such as at t_0 in Figure 5).

From the above statement, we only have to prove that

$$\sum_{i \in J_h} (T_i^h + E_i^h) \geq \sum_{k \geq 2} (k-1)t_k$$

in an arbitrary subset of jobs, J_h, divided as mentioned above. We first prove that for any sequence (1),(2),$\cdots$,(r) of jobs in J_h

$$\min r(\bar{T} + \bar{E}) \geq p_{(2)} + p_{(3)} + \cdots + p_{(r)} - (d_{(r)} - d_{(1)}) .$$

Consider jobs (2),$\cdots$,(r) as one job A with processing time $\sum_{i=2}^{r} p_{(i)}$, and due date $d_{(r)}$. Then T+E for jobs (1),(2),$\cdots$,(r) is not less than T+E for jobs (1) and A. This is because T+E for jobs (2),$\cdots$,(r-1) are all nonnegative and T+E for job (r) is equal to T+E for job A. By Corollary 4, for two adjacent jobs (1) and A, a schedule where either $C_{(1)} = d_{(1)}$ or $C_A = d_{(r)}$ is optimal. In both cases T+E for jobs (1) and A is not less than $\sum_{i=2}^{r} p_{(i)} - (d_{(r)} - d_{(1)})$. Next we prove that

$$TE \equiv p_{(2)} + \cdots + p_{(r)} - (d_{(r)} - d_{(1)}) \geq \sum_{k=2}^{r} (k-1)t_k .$$

Since there is no idle time, $\sum_{k=1}^{r} t_k \geq p_{(1)} + (d_{(r)} - d_{(1)})$.

Then

$$\begin{aligned} TE &= \sum_{k=1}^{r} p_{(k)} - p_{(1)} - (d_{(r)} - d_{(1)}) \\ &= \sum_{k=1}^{r} kt_k - p_{(1)} - (d_{(r)} - d_{(1)}) \\ &= \sum_{k=2}^{r} (k-1)t_k + \sum_{k=1}^{r} t_k - p_{(1)} - (d_{(r)} - d_{(1)}) \\ &\geq \sum_{k=2}^{r} (k-1)t_k, \qquad \text{since } \sum_{k=1}^{r} t_k \geq p_{(1)} + (d_{(r)} - d_{(1)}). \end{aligned}$$

The above is true for every sequence of r jobs considered.

This completes the proof. ∎

The above property is described pictorially in Figure 5, where there are two subsets divided by time t_0. Conceptually, Lemma 6 says that jobs must be shifted far enough forward or backward in time from their respective due dates to satisfy the constraint of at most one job on the machine at a given time. The lower bound on T+E gives the minimal amount of shifting to accomplish this on the assumption that each subset of jobs can be considered separately. In general, shifting in one subset will affect other subsets, so that the given bound may not be achievable.

If TG and s which result from Algorithm 1 are used, the lower bound B_2 can be improved as follows. Consider the case where (1) there are jobs not in PS whose due date is greater than s, and (2) TG resulting from the optimal timing for PS is greater than 0. The last k jobs (with largest due time) of J\PS and the optimal schedule for the jobs in PS incur T+E no less than $\sum_{i=1}^{k} (d_{<i>} - s)$, where <i> is the i-th job from the end in J\PS, and k is the minimum value of TG and n_s, the number of jobs in J\PS whose due date is greater than s. This bound can be justified as follows. If k = TG, either the k last jobs in J\PS should move to the left up to the time s, or jobs in PS should move to right, both of which incur T+E of no less than $\sum_{i=1}^{k} (d_{<i>} - s)$. Moreover in this case, Lemma 6 can be used to calculate T+E on the jobs not considered yet. Here the set of jobs in PS can be considered as a job whose due date is C_{PS}, the completion time of the jobs in PS, and processing time is $C_{PS} - s$. If $k = n_s$, k last jobs in J\PS should be move to the left, which incurs T+E of no less than $\sum_{i=1}^{k} (d_{<i>} - s)$.

Since the above algorithm requires exponential time in the worst case, we cannot guarantee that it is computationally tractable for larger problems. Therefore a heuristic algorithm is presented here. This algorithm uses the properties in Lemmas 2 and 5 to construct a good sequence, and uses Algorithm 1 to find an optimal timing for the sequence. By comparing all pairs of jobs using the results of Lemma 2, we can obtain rough information about the priority of each job, i.e., how many jobs should precede it. These priorities can give a good initial sequence which will be examined using dominance criteria of Lemma 5 and will be changed if needed.

Algorithm 3. (A heuristic algorithm)

Let α_i be the label of job i.

(1) Compare jobs i and j, for all possible combinations of 2 jobs, using the simple rule in Lemma 2.

If i precedes j, let $a_i = a_i - 1$, $a_j = a_j + 1$,

if j precedes i, let $a_i = a_i + 1$, $a_j = a_j - 1$.

else, no change in a_i, a_j.

(2) Sort the jobs in ascending order of a_i's. This will be the initial sequence.

(3) Obtain the optimal timing for the sequence resulted from (2) using Algorithm 1.

(4) Check the conditions for dominance in Lemma 5 for adjacent pairs of jobs, and change the order if needed. Check whether T+E can be reduced when the adjacent jobs are interchanged, and change the order if needed.

(5) For the sequence resulting from (4), obtain optimal timing using Algorithm 1. Terminate. ∎

At step (4), the pairwise comparison of interchange can be done in two directions, forward and backward. In forward comparisons, k-th job is compared with (k+1)th job, (k+2)th job, $\cdots$, until it cannot be changed any more, for k=n-1,n-2,$\cdots$,1. In backward comparisons k-th job is compared with (k-1)th job, (k-2)th job, $\cdots$, until it cannot be changed for k=2,3,$\cdots$,n. Since this requires $O(n^2)$ effort, and the other steps also require at most $O(n^2)$ effort, the overall computational complexity of this heuristic algorithm is $O(n^2)$.

3. $n/1//\Sigma(\tau_i T_i + \epsilon_i E_i)$ PROBLEM

In this section the general problem, $n/1//\Sigma(\tau_i T_i + \epsilon_i E_i)$ will be discussed. As in the $n/1//\bar{T}+\bar{E}$ problem, a special case of this problem, the optimal schedule(timing) for a given sequence can be obtained by solving an LP. Algorithm 1 can also be modified to solve this problem with $O(n^2 \log n)$ effort.

The needed modifications are:

a) In step (0), $G_n = 1$ should be changed to $G_n = \tau_n$;

b) In step (1), $G_j = -1$ should be changed to $G_j = -\epsilon_j$;

c) In step (2), $G_j = 1$ should be changed to $G_j = \tau_j$.

Getting an optimal sequence for this problem is even harder than for the case of $\tau_i = \epsilon_i = 1$, for all i. From Lemmas 2 and 5, and Corollaries 3 and 4, a loose lower bound for $\Sigma(\tau_i T_i + \epsilon_i E_i)$ can be presented. That is, a lower bound B_2 for jobs in a set J\PS is,

$$B_2 = \sum_{\omega} t_{\omega} \{ \sum_{i \epsilon \omega} \min(\tau_i, \epsilon_i) - \max_i [\min(\tau_i, \epsilon_i)] \} ,$$

where ω is any set of overlapping jobs in J\PS, and t_{ω} is the length of time during which jobs in ω overlap, if all the jobs in J\PS are placed as $C_i = d_i$. For example, in Figure 5, one of ω is (2,3,4) and $t_{(2,3,4)} = t_{32}$. Note that the terms in braces correspond to $k - 1$, and t_{ω} corresponds to t_k of Lemma 6. The lower bound B_1 for jobs in PS can be calculated by the same method as in Algorithm 2. A heuristic algorithm which will not be presented in this paper can be developed using a method similar to Algorithm 3.

4. COMPUTATIONAL RESULTS

A set of problems has been generated randomly for our experiment. The due dates follow a uniform distribution from 0 to a given maximum due date. The processing times have been generated from the uniform distribution such that the machine loads, i.e., the sum of these processing times divided by the maximum due date, would be between 0.6 and 1.3. The algorithms were coded in FORTRAN and run on the Amdahl 470V/8.

The results are given in Tables 1 and 2. Table 1 contains the problems in which machine load is greater than 0.9, while Table 2 contains those where the machine load is less than 0.9. As can be seen in the tables, when the machine load was high (≥ 0.9) the branch-and-bound algorithm could not solve the 30 job problems and could solve less than half of eight problems with 20 jobs. However, when machine load was low(< 0.9), it could solve 5 out of 8 problems with 30 jobs. When machine load is high; earliness rarely occurs, therefore the problem would look like a mean tardiness problem. From this point of view, our algorithm works better in the situations where earliness may be important, i.e., where machine load is not near 100%.

In the problems where the optimal solution was found by the branch-and-bound algorithm, over 90 percent of the solutions from the heuristic algorithm were optimal solutions. In most of the other problems, the heuristic solution is same as the incumbent solution obtained by the branch-and-bound algorithm after a CPU time of 30 seconds. In only two problems was the heuristic solution proved to be suboptimal. The minimum lower bound among the subproblems left in the stack is also given in the tables. Even though this minimum lower bound is better than the incumbent solution in some problems, it is likely to increase as more branching occurs. This difference between the minimum lower bound and the incumbent solution is higher in larger problems since the depth of the branch and bound tree was limited by CPU time. Thus, the minimum lower bounds may not be indicative of optimal objective values.

In summary, the heuristic procedure appears to provide optimal or near-optimal solutions in a fraction of the CPU time of the optimal procedure.

5. CONCLUSION

Scheduling decisions are generally affected by a number of costs. In this paper, scheduling problems which can be used in production and inventory planning have been discussed. The problem involving tardiness and earliness as dual criteria needs not only a sequence but also an optimal timing of the sequence. A polynomial algorithm has been developed for determining the optimal completion times of jobs for a given sequence. A branch-and-bound algorithm and a heuristic algorithm have been presented to obtain the schedule (sequence and timing). For a set of randomly generated problems, the heuristic algorithm performed extremely well.

There are two directions for future research on this problem. One is to develop a good(polynomial) algorithm for these problems. The other is to prove NP-completeness of the problem, $n/1//\bar{T}+\bar{E}$ with equal due dates, and to develop a more efficient B&B algorithm or an heuristic algorithm. In addition, extension of the results to multiple machine problems would be very useful for real applications. These problems would cover the cases where jobs have different penalties for earliness and tardiness, which occur frequently in reality.

REFERENCES

Bagchi, V., R.S. Sullivan, and Y.L. Chang, 1986. Minimizing Mean Absolute Deviation of Completion Times about a Common Due Date. *Naval Research Logistics Quarterly*, Vol.33, pp.227-240.

Baker, K.R., 1974. *Introduction to Sequencing and Scheduling.* Wiley, New York.

Emmons, H., 1975a. One Machine Sequencing to Minimize Mean Flow Time with Minimum Number Tardy. *Naval Research Logistics Quarterly*, Vol.22, No.3, pp.585-592.

Emmons, H., 1975b. A Note on a Scheduling Problem with Dual Criteria. *Naval Research Logistics Quarterly*, Vol.22, No.3, pp.615-616.

Garey, M.R. and D.S. Johnson, 1979. *Computers and Intractability A Guide to the Theory of NP-Completeness*, Freeman.

Gupta, S.K. and T. Sen, 1983. Minimizing a Quadratic Function of Job Lateness on a Single Machine. *Engineering Costs and Production Economics*, Vol.7, pp.187-194.

Kanet, J.J., 1981. Minimizing the Average Deviation of Job Completion Times about a Common Due Date. *Naval Research Logistics Quarterly*, Vol.28, No.4, pp.643-651.

Lakshiminarayan, S., R. Lakshmanan, R.L. Papineau, and R. Rochette, 1978. Optimal Single-Machine Scheduling with Earliness and Tardiness Penalties. *Operations Research*, Vol.26, No.4, pp.1079-1082.

Lin, K.S., 1983. Hybrid Algorithm for Sequencing with Bicriteria. *Journal of Optimization Theory and Applications*, Vol.39, No.1, pp.905-924.

Nelson, R.T., R.K. Sarin, and R.L. Daniels, 1986. Scheduling with Multiple Performance Measures: The One-Machine Case. *Management Science*, Vol.32, No.4, pp.464-479.

Panwalkar, S.S., M.L. Smith, and A. Seidmann, 1982. Common Due Date Assignment to Minimize Total Penalty for the One Machine Scheduling Problem. *Operations Research*, Vol.30, No.2, pp.391-399.

Sen, T. and S.K. Gupta, 1983. A Branch-and-Bound Procedure to Solve a Bicriterion Scheduling Problem. *IIE Transactions*, Vol.15, No.1, pp.84–88.

Sen, T. and S.K. Gupta, 1984. A State-of-Art Survey of Static Scheduling Research Involving Due Dates. *OMEGA The Int. Jl. of Mgmt. Sci.* Vol.12, No.1, pp.63–76.

Sidney, J.B., 1977. Optimal Single Machine Scheduling with Earliness and Tardiness Penalties. *Operations Research*, Vol.25, No.1, pp.62–69.

Sundararaghavan, P.S. and M.U. Ahmed, 1984. Minimizing the Sum of Absolute Lateness in Single-Machine and Multimachine Scheduling. *Naval Research Logistics Quarterly*, Vol.31, pp.325–333.

Townsend, W., 1978. The Single Machine Problem with Quadratic Penalty Function of Completion Times: A Branch-and-Bound Solution. *Management Science*, Vol.24, No.5, pp.530–534.

Van Wassenhove, L.N. and L.F. Gelders, 1980. Solving a Bicriterion Scheduling Problem. *European Journal of Operational Research*, Vol.4, No.1, pp.42–48.

Van Wassenhove, L.N. and K.R. Baker, 1982. A Bicriterion Approach to Time/Cost Trade-offs in Sequencing. *European Journal of Operational Research*, Vol.11, No.1, pp.48–54.

Table 1. Computational Results. (Machine load is ≥ .9)

No. of jobs	Heuristic		Branch and Bound			
	Solution	CPU time *	Solution	CPU time *	# of sub-problems	min lower bound
10	188	.002	188	1.016	1653	
	112	.002	112	.089	92	
	225	.003	225	.786	1305	
	95	.002	95	.206	315	
15	327	.004	327 **	> 30.	16345	241
	316	.004	316 **	> 30.	17238	241
	220	.004	220	31.876 †	30124	
	288	.004	288	.601	517	
20	131	.007	131 **	> 30.	18657	120
	110	.006	110 **	> 30.	17099	76
	110	.006	110	5.438	4354	
	75	.006	75 **	> 30.	20075	55
	69	.006	69	2.313	1846	
	113	.006	113 **	> 30.	15325	74
	278	.006	278 **	> 30.	16068	192
	257	.005	257	56.549 †	44417	
30	209	.010	209 **	> 30.	12584	63
	174	.010	174 **	> 30.	13134	122
	732	.012	732 **	> 30.	12396	289
	386	.011	386 **	> 30.	10686	184

* CPU time in seconds on Amdahl 470V/8 system

** indicates that it is not verified as optimal (it is current incumbent solution)

† These problems were run for more than 30 seconds to get an optimal solution since the solution obtained after 30 seconds was very close to the lower bound.

Table 2. Computational Results. (Machine load is ≤ .9)

No. of jobs	Heuristic		Branch and Bound			
	Sólution	CPU time *	Solution	CPU time *	# of sub-problems	min lower bound
10	48	.002	48	.074	66	
	30	.002	30	.223	108	
	51	.002	51	.318	289	
	59	.002	59	.205	48	
15	35	.003	35	.187	194	
	53 ***	.003	51	.590	473	
20	18	.005	18	.554	437	
	188	.005	188	37.826 †	30875	
	120	.005	120	.779	609	
	124 ***	.005	123	.933	776	
	130	.006	130	7.953	6139	
	99	.005	99	4.906	3926	
	127	.005	127	12.443	10223	
	354	.006	354 **	> 30.	19330	269
30	61	.009	61	2.819	1216	
	91	.009	91	1.757	841	
	67	.009	67 **	> 30.	13819	48
	152	.009	152 **	> 30.	13900	107
	158	.009	158	19.723	10060	
	121	.009	121	8.911	4609	
	119	.010	119	4.924	2108	
	303	.010	303 **	> 30.	14590	188
40	100	.015	100 **	> 30.	8610	47
	216	.015	216 **	> 30.	8654	176
	659	.016	659 **	> 30.	8105	225
	471	.016	471 **	> 30.	7510	192

* CPU time in seconds on Amdahl 470V/8 system

** indicates that it is not verified as optimal solution (current incumbent solution)

*** indicates that it is not equal to optimal solution

† This problem was run for more than 30 seconds to get an optimal solution.

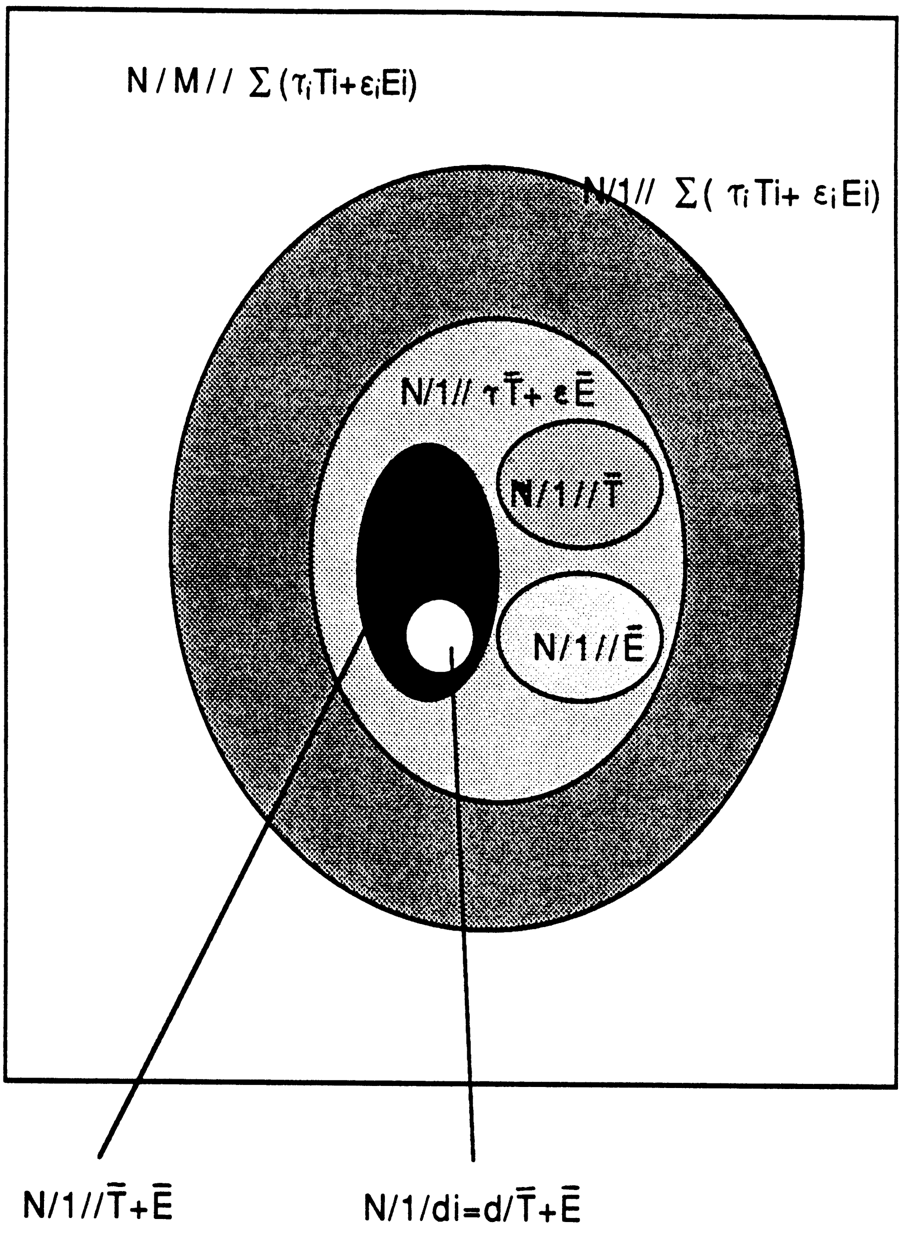


Figure 1. Relationships among the problems

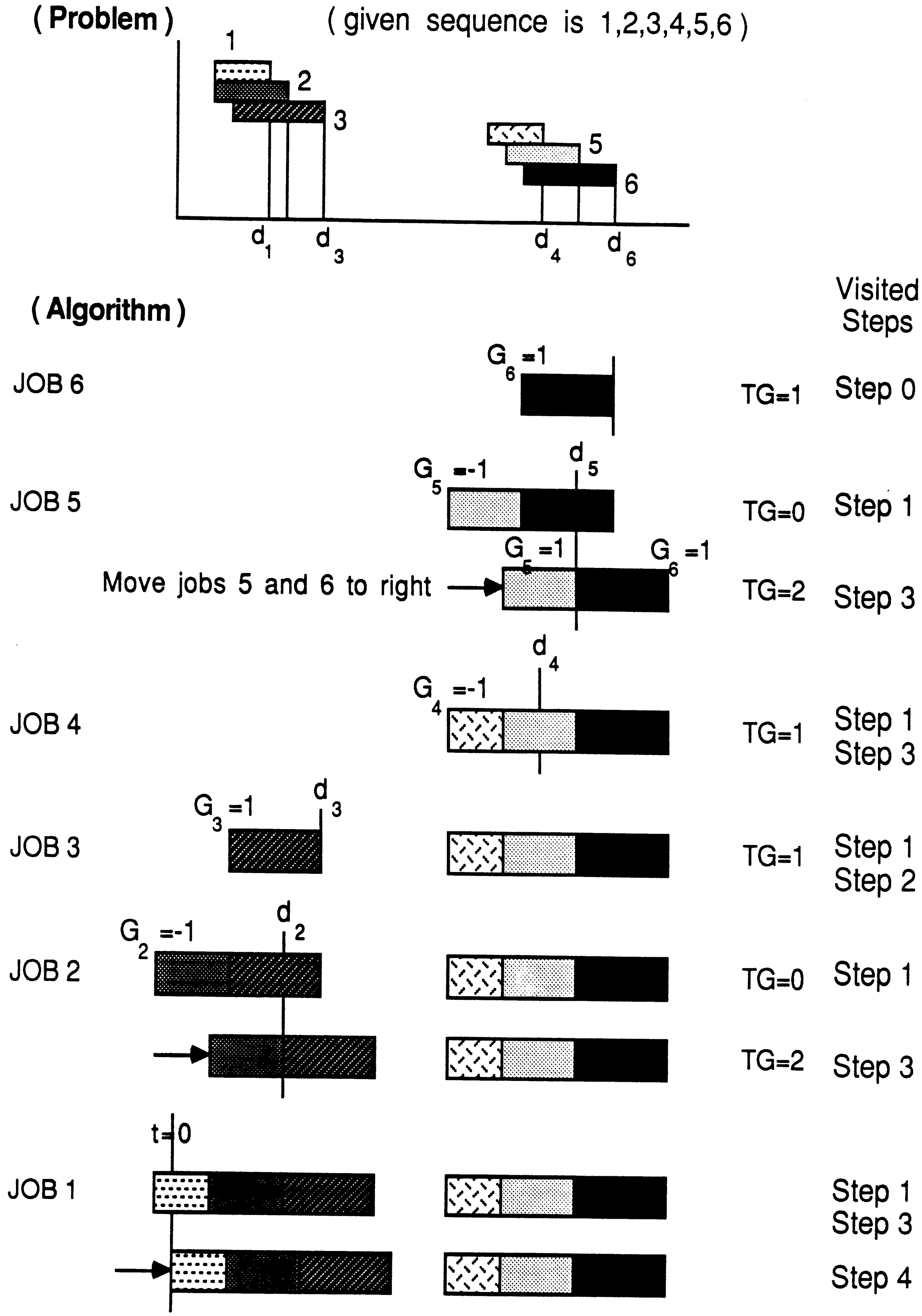
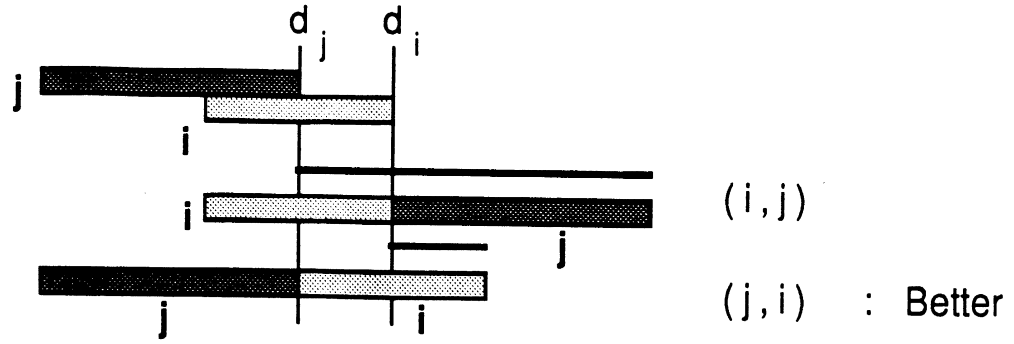
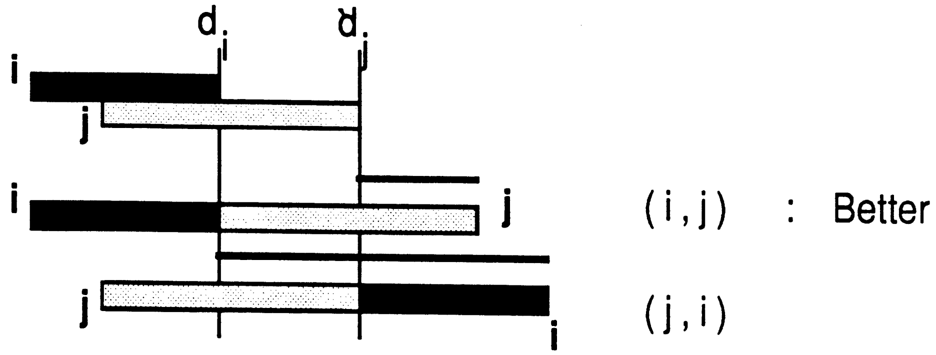


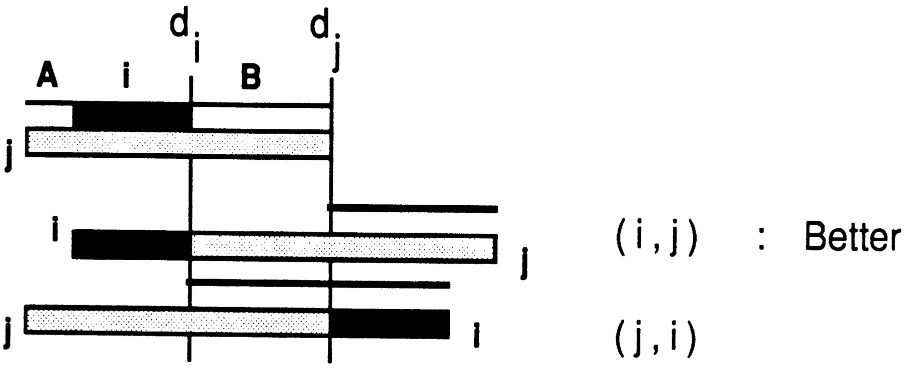
Figure 2. An example for Algorithm 1.



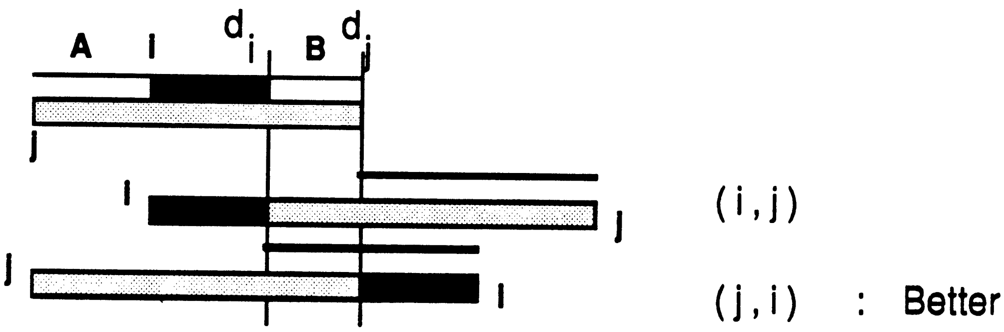
(a) $d_i > d_j$



(b) $d_i < d_j$ and $p_i + (d_j - d_i) > p_j$

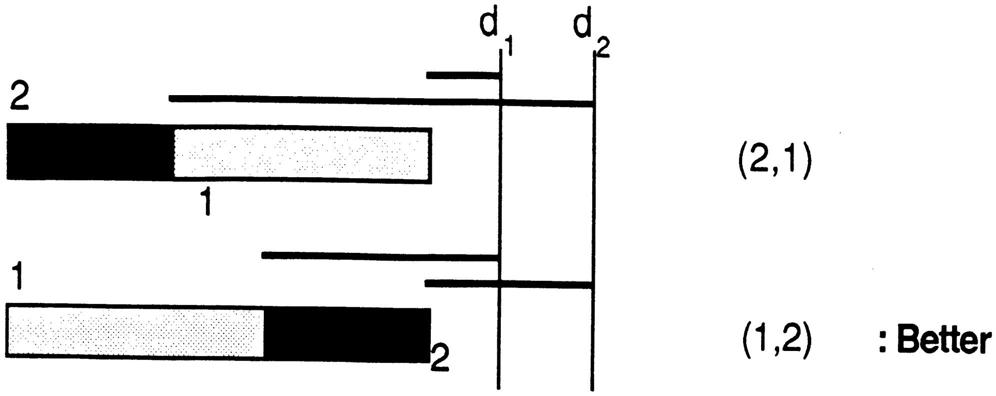


(c) $d_i < d_j$, $p_i + (d_j - d_i) < p_j$, and $A < B$

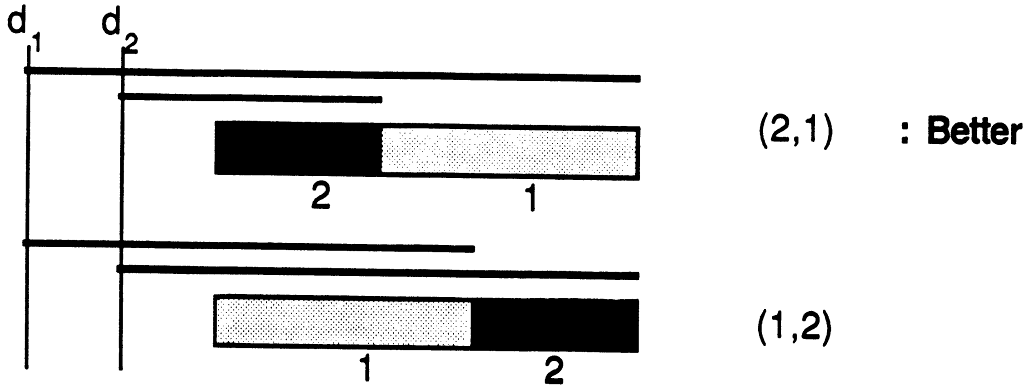


(d) $d_i < d_j$, $p_i + (d_j - d_i) < p_j$, and $A > B$

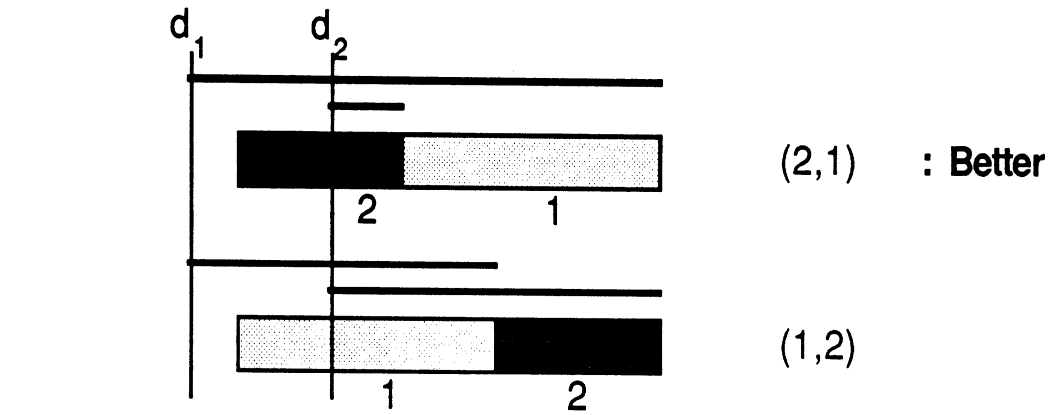
Figure 3. Pictorial view for Lemma 2.
(Solid lines denote T+E)



(a) $\max[C_1, C_2] \le \min[d_1, d_2]$: LPT

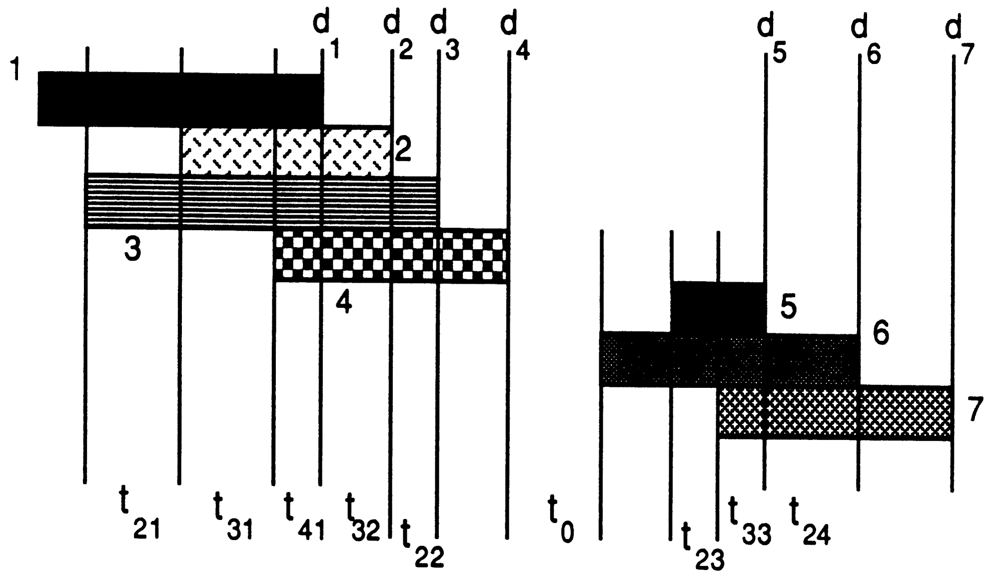


(b) $\min[s_1, s_2] \ge \max[d_1, d_2]$: SPT



(c) $\min[s_1, s_2] + \min[p_1, p_2] \ge \max[d_1, d_2]$: SPT

Figure 4. Properties for pruning. (Lemma 5)
(Solid lines denote T+E for 2 jobs.)



$$B_2 = (t_{21} + t_{22} + t_{23} + t_{24}) + 2(t_{31} + t_{32} + t_{33}) + 3\,t_{41}$$

Figure 5. Lower bound (B_2) for T+E of the jobs in J\PS .