SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK 14853–7501

TECHNICAL REPORT NO. 768

January 1988
Revised July 1989

WEIGHTED–TARDINESS SCHEDULING ON PARALLEL MACHINES WITH PROPORTIONAL WEIGHTS

by

by Esther M. Arkin and Robin O. Roundy

This research was supported by NSF Grant DMC–8451984 and also by AT&T Information Systems and DuPont Corporation.

WEIGHTED–TARDINESS SCHEDULING ON PARALLEL MACHINES WITH PROPORTIONAL WEIGHTS

by Esther M. Arkin and Robin O. Roundy

Abstract

In this paper we address the problem of scheduling a number of jobs on a bank of parallel machines so as to minimize the total weighted tardiness, under the assumption that the weight of each job is proportional to its processing time. The version of the problem that has general weights has been shown to be strongly NP-complete. We prove this version of the problem to be NP-complete, and give a pseudo-polynomial time algorithm for solving it. We study a family of simple sequencing rules in which the jobs are sequenced in increasing order of $\gamma_i = d_i - \theta p_i$ where d_i is the due date of job i, p_i is its processing time, w_i is its weight, and $0 \leq \theta \leq 1$. This family of sequencing rules generalizes the earliest-due-date (EDD) sequencing rule. We obtain bounds on the ratio $[C_\gamma - C^*]/[\Sigma_i\ w_i\, p_i]$ where C_γ and C^* are the costs of the heuristic and optimal schedules, respectively. The denominator is the cost of having each job be late by its own processing time. It is intended to measure what is or is not a large deviation from optimality, in absolute terms rather than relative terms, for the problem at hand. We also report on the results of computational experiments.

WEIGHTED–TARDINESS SCHEDULING ON PARALLEL MACHINES WITH PROPORTIONAL WEIGHTS.

by Esther M. Arkin and Robin O. Roundy

In this paper we address the problem of scheduling a number of jobs on a bank of parallel machines so as to minimize the total weighted tardiness, under the assumption that the weight of each job is proportional to its processing time. The version of the problem that has general weights has been shown to be strongly NP-complete. We prove this version of the problem to be NP-complete, and give a pseudo-polynomial time algorithm for solving it. In addition we study a family of very simple sequencing rules. Given a value of θ, $0 \leq \theta \leq 1$, we set $\gamma_i = d_i - \theta p_i$ where d_i is the due date of job i and p_i is its processing time. We study *earliest-gamma-date* (EγD) sequencing rules, in which the jobs are sequenced in increasing order of γ_i. The family of EγD sequencing rules generalizes the earliest-due-date (EDD) sequencing rule (for $\theta = 0$).

The worst–case relative cost of the EγD sequencing rules is infinite. However large relative costs occur only when the cost of an optimal sequence is zero or near-zero, and the cost of an earliest-gamma-date sequence is small. In this paper we obtain bounds on the ratio $[C_\gamma - C^*]/[\Sigma_i w_i p_i]$ where w_i is the weight of job i, and C_γ and C^* are the costs of the heuristic and optimal schedules, respectively. The denominator is the cost of having each job be late by its own processing time. It is intended to measure what is or is not a large deviation from optimality, in absolute terms rather than relative terms, for the problem at hand. We also report on the results of computational experiments.

We were motivated to do this research when we observed EDD sequences being used very effectively with the weighted tardiness objective function in real-world scheduling systems, in spite of the fact that EDD sequences do not perform well for general weighted tardiness problems. In light of this research we conclude that EDD and other EγD

sequencing rules work very well for weighted-tardiness scheduling problems when the weights of the jobs are roughly proportional to the job processing times. They also work well when very little tardiness occurs. But they do not work well otherwise.

In practice appropriate weights for jobs are difficult to ascertain. In real-world scheduling systems the weights are often set arbitrarily, in many cases by setting all of them equal to one. An alternative approach would be to use weights that are proportional to the processing times of the jobs. Typically jobs that have longer processing times are larger jobs that have a higher selling price and, consequently, a higher priority. In many systems both the processing time of a job and the weight of a job are nearly proportional to the job's dollar value, resulting in weights that are nearly proportional to the processing times.

When studying the deviation of the cost of a heuristic algorithm from optimality, careful attention should be paid to the benchmark against which the deviation is compared, in this case, $[\Sigma_i w_i p_i]$. We list a few properties which such a benchmark should have. First, it should have the same units as the objective function – in this case, dollars. Second, it should be insensitive to the addition of dummy jobs, with zero weights and zero processing times, that have no effect on the essential nature of the scheduling problem. Third, suppose that a problem B consisted of n independent instances of a smaller problem A, strung together over time in such a way that the costs of both the optimal and the heuristic solutions to problem B are n times as large as the respective costs for problem A. Then the benchmark for B should be n times as large as the benchmark for A.

Up to an order of magnitude, the benchmark $\Sigma_i w_i p_i$ against which we compare the absolute deviation from optimality is the smallest function of the weights and the processing times that we could devise that has these properties. For example, if $\bar{p}$ is the average processing time then $\Sigma_i w_i \bar{p}$ violates the second property and $max_i w_i p_i$ violates

the third property. Also note that the cost of an optimal policy can be much larger than the value of the benchmark $\Sigma_i w_i p_i$.

To our knowledge, the weighted-tardiness problem with weights proportional to the processing times has never been addressed. The general weighted tardiness problem is strongly NP-Complete (Lawler 1977, Lenstra et al 1977). Two special cases are known to be more tractable. If the processing times are all equal, the problem reduces to the classical assignment problem, so it can be solved in polynomial time. Lawler (1977) has developed a pseudo-polynomial time algorithm for weighted tardiness problems with agreeable weights, i.e., for systems in which $p_i > p_j$ implies that the weight for i is less than or equal to the weight for j. Intuitively one would expect the opposite, i.e., jobs with longer processing times should be more important than jobs with shorter processing times. Lawler's algorithm clearly applies to problems in which all of the weights are equal. The equal-weight case has recently been shown to be NP-hard (Du and Leung 1988).

Numerous heuristics have been developed to solve the general weighted tardiness problem (Morton et al 1984, Fisher 1976), and dynamic programming algorithms (Baker and Schrage 1978a,b, Lawler 1979, Kao and Queyranne 1982) and branch and bound algorithms (Fisher 1976) for computing optimal solutions to it have been proposed. Also see (Lenstra et al 1982, Lenstra et al 1985).

Although the weighted-tardiness problem with weights proportional to the processing times has apparently never been addressed, there are a number of papers on the weighted completion time problem with weights proportional to the processing times. See for example Eastman, Even and Isaacs (1964), Kawaguchi and Kyan (1986), and Weiss (1988a, 1988b).

The remainder of this paper is organized as follows. In section 1 we formally state our assumptions and give several general properties for single-machine, weighted-tardiness scheduling problems in which the weights are proportional to the processing times.

In section 2 we derive a lower bound on the cost of an optimal schedule for a one-machine problem. The worst-case analysis for one machine problems is found is section 3, and the results are extended to multi-machine problems in section 4. Computational results for one-machine problems are reported in section 5. In section 6 we present some important properties of optimal sequences for weighted-tardiness problems where the weights are proportional to the processing times, and we give a pseudo-polynomial time algorithm for computing an optimal sequence. A proof that the weighted-tardiness problem where the weights are proportional to the processing times is NP–hard appears in section 7. Section 8 contains our conclusions and some observations regarding future research. The paper is written in such a way that readers who are interested primarily in the complexity issues can skip sections $2-5$.

SECTION 1: ASSUMPTIONS, NOTATION, AND GENERAL PROPERTIES.

In this section we formally state our assumptions and the notation we will use. We then derive a number of special properties of single-machine, weighted-tardiness scheduling problems in which the weights are proportional to the processing times, which distinguish this special case of the weighted tardiness scheduling problem from the more general version. These properties will be used in the sequel.

A single machine is available for processing continuously, starting at time B. We are given a finite job set J, all of which are available for processing at time B. Each job $i \in J$ has a processing time $p_i > 0$ and a due date $d_i > B$. We assume without loss of generality that the weight of job i is equal to its processing time, i.e., $w_i = p_i$, which can be accomplished by a change of units. The jobs in J must be processed on the machine without pre-emption. The machine is capable of processing only one job at a time. The *tardiness* of a job is the amount by which its completion time exceeds its due date. Algebraically, it can be written as $w_i(c_i - d_i)^+$ where w_i is the weight of job i, c_i is its completion time, d_i is its due date, and x^+ denotes the positive part of x. The

objective is to find a schedule which minimizes the weighted sum of the tardiness of the jobs, under the assumption that $w_i = p_i$ for all $i \in J$.

Note that there is no incentive to idle the machine. Therefore we can assume that the machine works without interruption from time B until time $F \equiv B + \Sigma_i p_i$. The start times and the completion times of all of the jobs are determined by the sequence in which the jobs are processed. Therefore we will use the terms "schedule" and "sequence" interchangeably.

Our notation is listed below.

NOTATION.

$\mathcal{J}$: The set of jobs.

J : The number of jobs in $\mathcal{J}$.

i, j : job indices. $1 \leq i, j \leq J$.

d_i : The due date for job i.

p_i : The processing time of job i.

$b_i = d_i - p_i$: The begin date for job i, the time by which processing on job i must begin if the job is to be completed on time.

B : The time at which the machine will initially become available for use.

$F = B + \Sigma_i p_i$: The time at which the last job is completed.

P : The problem of scheduling the jobs in $\mathcal{J}$ starting at time B so as to minimize the total weighted tardiness, assuming that the weight of each job is equal to its processing time.

S : A schedule for the jobs in $\mathcal{J}$.

$c_i(S)$: The completion time of job i in schedule S.

$\mathcal{E} = \{i : c_i(S) \leq d_i\}$: The set of jobs that are either early or exactly on time in schedule S.

$\mathcal{L} = \{i : c_i(S) > d_i\}$: The set of (strictly) late jobs in schedule S.

$x \wedge y$: The minimum of the two numbers x and y.

$x \vee y$: The maximum of the two numbers x and y.

x^+ : $x \vee 0$, or the positive part of x.

$C(S) = \Sigma_i\, p_i (c_i(S) - d_i)^+$: The cost of the sequence S.

LB : A Lower Bound on the Cost of an Optimal Schedule for P .

θ : A user-specified constant satisfying $0 \leq \theta \leq 1$.

$\bar{\theta} = 1 - \theta$.

$\gamma_i = \theta b_i + \bar{\theta} d_i = d_i - \theta p_i = b_i + \bar{\theta} p_i$: The gamma-date of job i.

Since no job can start before time B and no job will complete after time F, we assume without loss of generality that

$$B \leq b_i \quad \text{and} \quad d_i \leq F \quad \text{for all jobs} \quad i. \tag{1}$$

If $b_i < B$, we can get an equivalent problem by setting $b_i = B$ and adding an appropriate constant to the cost.

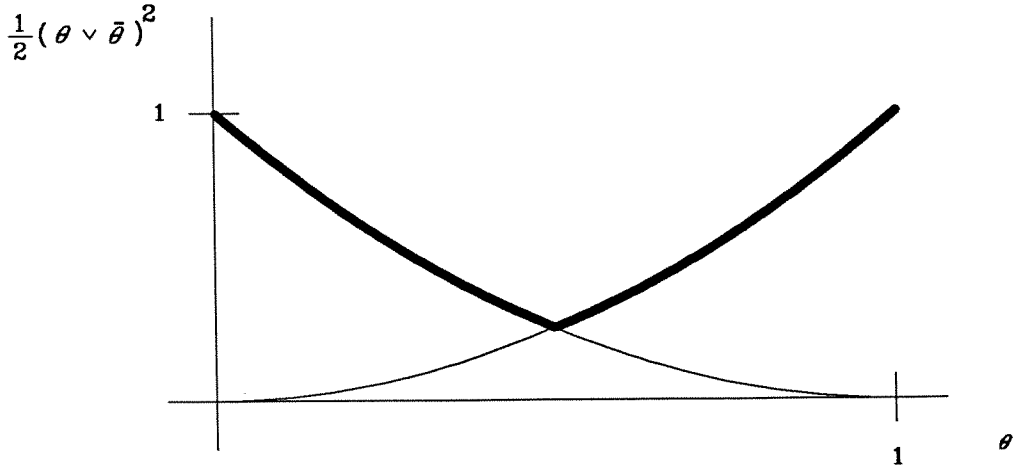
One of our major results for one-machine systems is the following.

Theorem 1. The cost of an EγD schedule can not exceed the cost of an optimal schedule by more than $\frac{1}{2}(\theta \vee \bar{\theta})^2 \Sigma_i\, p_i^2$ where $\gamma_i = b_i + \bar{\theta} p_i$.

This bound is illustrated in Figure 1 as a function of θ. Note that the best value of θ is one half, which gives a bound of $\frac{1}{8} \Sigma_i\, p_i^2$. This corresponds to sequencing jobs in increasing order of their half-dates, the average of their due dates and their begin dates.

The proof of Theorem 1 is found in section 4. It is based on some general properties of one-machine scheduling problems with weights equal to the processing times. We now explore these properties, several of which are of interest in their own right.

Figure 1: The Bound as a Function of θ



In a single-machine weighted tardiness problem, if all of the due dates are less than or equal to B then all of the jobs are going to be late in any schedule. In this case the weighted tardiness problem reduces to the problem of minimizing the weighted completion time, $\Sigma_i w_i c_i(S)$. It is well known that the optimal schedules for this problem are the weighted shortest processing time (WSPT) sequences, in which jobs with smaller values of p_i/w_i go first and ties are broken arbitrarily. However if $p_i = w_i$ then all sequences are WSPT. This is confirmed by the fact that if $S = \{1, 2, ..., J\}$ then

$$\sum_i p_i c_i(S) = \sum_i p_i \left[B + \sum_{j \leq i} p_j \right]$$

$$= \frac{1}{2}\left[\sum_i p_i \right]^2 + \frac{1}{2}\sum_i p_i^2 + \sum_i p_i B ,$$

which is sequence-independent. We will make frequent use of this algebraic reduction.

In a general single-machine weighted tardiness scheduling problem, if there is an optimal schedule in which no jobs are late then an EDD sequence is optimal (Conway et al 1967). As we mentioned in the preceding paragraph, if the due dates of all of the jobs

are already past then a WSPT sequence is optimal. The weighted tardiness problem can be thought of as as the problem of finding an optimal compromise between an EDD sequence and a WSPT sequence. When the weights are proportional to the processing times the EDD sequence is WSPT, so one would expect EDD to be a very good sequence. We will prove that this is in fact the case. It is interesting to note that the other case in which a pseudopolynomial time algorithm for the weighted tardiness problem exists, the case of agreeable weights, also has the property that the EDD sequence is WSPT.

The earliness of job i in schedule S is $(d_i - c_i(S))^+$, and the weighted earliness of schedule S is $E(S) = \Sigma_i\, p_i(d_i - c_i(S))^+$. The following lemma implies that minimizing the weighted tardiness and minimizing the weighted earliness are equivalent, if the weights are equal to the processing times.

Lemma 2. The total weighted earliness of S and the total weighted tardiness of S differ by $\frac{1}{2}\left[\Sigma_i\, p_i\right]^2 + \frac{1}{2}\Sigma_i\, p_i^2 + \Sigma_i\, p_i(B - d_i)$, which is sequence-independent.

Proof. Assume that the jobs are indexed so that $S = \{1,2,\ldots, J\}$. Then

$$\begin{aligned} C(S) - E(S) &= \sum_i \left[\, p_i\,(c_i(S) - d_i)^+ - p_i\,(d_i - c_i(S))^+ \,\right] \\ &= \sum_i p_i\,(c_i(S) - d_i) = \sum_i p_i \left[\, B + \sum_{j \le i} p_j \,\right] - \sum_i p_i d_i \\ &= \frac{1}{2}\left[\, \sum_i p_i \,\right]^2 + \frac{1}{2}\sum_i p_i^2 + \sum_i p_i(B - d_i)\,. \quad \square \end{aligned}$$

Let Q be the original scheduling problem. Note that the completion time of the last job in a schedule for problem Q that has no idle time will be $B + \Sigma_i\, p_i$. Consider the scheduling problem RQ, which is the reversal of problem Q in time. The values of B and F, the job set, the weights, and processing times for RQ are the same as they are

for Q . The due date for job i in problem RQ is $B + (F - d_i + p_i)$. If $S = \{1, 2, ..., J\}$ is a schedule for Q then the *reversal* of S is $\{J, J-1, ..., 1\}$.

For a given schedule S let S^* be the reversal of S , let $C^*(S^*)$ be the cost of S^* in problem RQ , and let $c_i^*(S^*)$ be the completion time of job i in S^* . Then $c_i^*(S^*) = B + (F - c_i(S) + p_i)$, so the tardiness of job i in S^* is $(d_i - c_i(S))^+$, the earliness of i in S. Therefore the weighted tardiness of S^* is equal to the weighted earliness of S. Lemma 2 implies the following.

COROLLARY 3. Problems Q and RQ are equivalent in the following sense. For every instance of Q , the difference between the cost of the schedule S for problem Q and the cost of the reversal of S for problem RQ is $\frac{1}{2}\left[\Sigma_i p_i\right]^2 + \frac{1}{2}\Sigma_i p_i^2 + \Sigma_i p_i(B - d_i)$, which is sequence-independent.

COROLLARY 4. If none of the jobs in a schedule is early then the schedule is optimal.

PROOF. Let S be the schedule and let S^* be its reversal. Then S^* has no late jobs, so it is optimal for problem RQ . By Lemma 2, S must be optimal for problem Q . □

In an EBD sequence jobs are scheduled in increasing order of their begin dates b_i . The EBD sequence is a special case of the EγD sequences, with $\theta = 1$.

COROLLARY 5. If a consecutive block of early jobs is re-sequenced into an EDD sequence, the new sequence has no late jobs and is therefore not more expensive than the original. If a consecutive block of late jobs is re-sequenced into an EBD sequence, the new sequence has no early jobs and thus is not more expensive than the original.

PROOF. The first assertion is a direct consequence of the well-known fact that the EDD sequence minimizes the maximum lateness for a single-machine scheduling problem (Conway et al 1967). The second assertion follows from the first and from Lemma 2. □

For general weighted tardiness problems, it is often the case that simple rules are able to provide a guarantee that there is an optimal schedule in which one specific job precedes another. See, for example, (Emmons 1969) and (Fisher 1976). If the weights are proportional to the processing times the following rule is obtained, which is much simpler and much more powerful than the rules for the general case.

LEMMA 6. If there is a time T and a pair of sets of jobs $\mathcal{K}$, $\mathcal{L}$ such that $\mathcal{J} = \mathcal{K} \cup \mathcal{L}$, $\mathcal{K} \cap \mathcal{L} = \phi$, $d_k \leq T$ for all $k \in \mathcal{K}$, and $b_\ell \geq T$ for all $\ell \in \mathcal{L}$, then there is an optimal schedule in which all jobs in $\mathcal{K}$ precede all jobs in $\mathcal{L}$.

PROOF. Let S^* be a schedule for which the hypothesis of Lemma 6 does not hold. Then in S^* there is a job $\ell \in \mathcal{L}$ that directly precedes a job $k \in \mathcal{K}$. Let S be the schedule obtained by interchanging ℓ and k in S. If job k completes at time t in S^* then

$$C(S) = A + p_k(t - p_\ell - d_k)^+ + p_\ell(t - d_\ell)^+ \quad \text{and}$$

$$C(S^*) = A + p_k(t - d_k)^+ + p_\ell(t - p_k - d_\ell)^+$$

where A is the cost associated with jobs other than k and ℓ. Therefore $C(S) - C(S^*) = f(t - p_\ell)$ where

$$f(x) = p_k[(x - d_k)^+ - (x + p_\ell - d_k)^+] + p_\ell[(x - b_\ell)^+ - (x - p_k - b_\ell)^+].$$

It suffices to show that $f(x) \leq 0$ for all x. Let $\tau = (d_k + b_\ell)/2$. Then $d_k \leq \tau \leq b_\ell$.

Claim 1: $f(x) = 0$ whenever $|x|$ is sufficiently large. *Proof*: Evident.

Claim 2: $f(\tau) < 0$. If $d_k < b_\ell$ then $f'(\tau) = 0$. *Proof*: If $d_k \leq t \leq b_\ell$ then $f(t) = p_k[(t - d_k) - (t + p_\ell - d_k)] + p_\ell[0 - 0] = -p_k p_\ell < 0$, independent of t.

If $d_k = b_\ell$ then f is piece-wise linear with three discontinuities. Claims 1 and 2 imply that $f(x) \leq 0$ for all x. If $d_k < b_\ell$ then f is piece-wise linear with four discontinuities, at $d_k - p_\ell$, d_k, b_ℓ, and $b_\ell + p_k$. Claim 2 implies that $f'(\tau) = 0$ and $f(\tau) < 0$, so by Claim 1, $f(x) \leq 0$ for all x. □

A *cluster* is a maximal set of jobs that has the following property: For all partitions $(\mathcal{K}, \mathcal{L})$ that satisfy the conditions of Lemma 6, the set in question is either a subset of $\mathcal{K}$ or a subset of $\mathcal{L}$. The set of all clusters is a partition of the set of jobs $\mathcal{J}$. Let $\mathcal{M}$ be a cluster, $b = min_{i \in \mathcal{M}} b_i$ be the earliest begin date in the cluster, $d = max_{i \in \mathcal{M}} d_i$ be the latest due date in the cluster, and $b < T < d$. Then there must be a job i such that $b_i < T < d_i$, and i must belong to $\mathcal{M}$. Otherwise $\mathcal{M}$ would not have the required property.

Lemma 6 implies that an optimal solution to the one-machine scheduling problem Q can be found by finding optimal sequences for each of the clusters and then piecing them together. The EγD schedules also sequence the jobs by cluster. In fact, if Theorem 1 can be shown to hold for problems Q with only one cluster then Lemma 6 can be used to show that it holds in general. Consequently most of the analysis in the sequel is done for a single cluster.

In decomposing a scheduling problem into a smaller problems, one for each cluster, it is important to select the value of B for each sub-problem properly. The obvious rule is the correct one. For a given cluster, the correct value of B is the time at which the first job in the cluster begins processing, assuming that there is no unnecessary idle time and that the jobs are sequenced cluster by cluster, according to Lemma 6.

For completeness, we give an algorithm for determining the clusters. First sort the set of all due dates and all begin dates for all of the jobs. In the event of a tie, put due dates before begin dates in the sorted list. Then scan the list, keeping track of the difference between the number of begin dates encountered so far and the number of due dates encountered so far. Whenever this difference becomes zero, a new cluster has been

identified. The cluster consists of the set of jobs whose due dates were encountered after the previous time when the difference became zero. This algorithm can be implemented to run in $O(J \log J)$ time.

SECTION 2: A LOWER BOUND.

In this section we derive a lower bound on the cost of an optimal policy for a one-machine scheduling problem with proportional weights. This lower bound will be used to prove Theorem 1. The lower bound applies to problems with only one cluster. For multi-cluster problems, the sum of the lower bounds of one-cluster problems gives a lower bound for the original problem, by Lemma 6.

No job can start before time B and no job will complete after time F. Therefore if $d_i > F$ then we can set $d_i = F$ without effecting the cost of any schedule. If $b_i < B$ then we can set $b_i = B$. This will add the constant $p_i(B - b_i)$ to the cost of every schedule. Therefore we can and do assume that

$$B \leq b_i \quad \text{and} \quad d_i \leq F \text{ for all jobs } i. \tag{1}$$

Unless otherwise stated, in section 2 and section 3 we assume that (1) holds and that Q has a single cluster.

Let

$$g(x) = \sum_i [(x - b_i)^+ \wedge p_i],$$

and let

$$z = inf\{x > B : g(x) \geq x - B\}. \tag{2}$$

Recall that $B \leq b_i$ and $d_i \leq F$ for all jobs i. Since $\Sigma_i\, p_i = g(F) = F - B$, z is well–defined. Since $g(B) = 0$ and g is continuous,

$$g(z) = z - B \quad \text{and} \quad B \leq z \leq F .$$

The definition of z is illustrated in Figure 2. If preemption were allowed, the amount of time required to complete the work corresponding to the shaded regions would be $z - B$. This fact is the key to our lower bound.

We obtain a lower bound by defining a new scheduling problem called LBQ. The cost of an optimal solution to LBQ is an easily-computed lower bound on the cost of an optimal solution to Q. We define LBQ by subdividing each job i in Q into two different jobs, job $i1$ and job $i2$. These new jobs satisfy

$$b_{i1} = b_i ,\ d_{i2} = d_i ,\ d_{i1} = b_{i2} = max\,(b_i ,\, min(z ,\, d_i)) ,$$

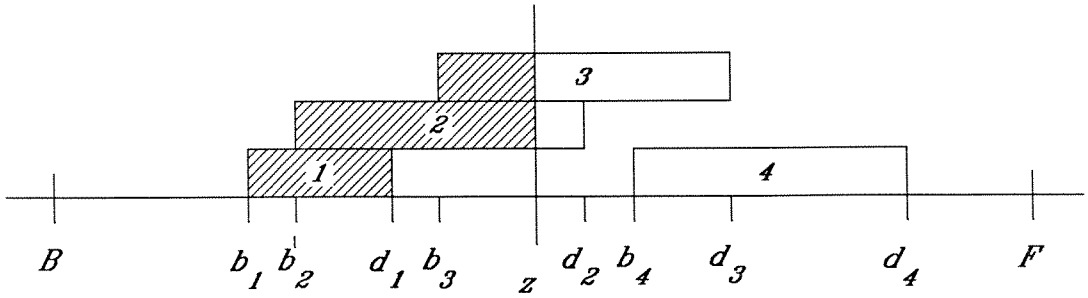
$$p_{i1} = d_{i1} - b_{i1} , \text{ and } p_{i2} = d_{i2} - b_{i2} . \tag{3}$$

The weights of the new jobs are equal to their processing times. Figure 3 illustrates. There may be new jobs that will have processing times of zero (jobs 12 and 41 in Figure 3). These jobs will have no impact on the solution to LBQ, but they are retained for notational convenience. Some simple, useful facts are

$$z - B = \sum\nolimits_i p_{i1} \quad \text{and} \tag{4}$$

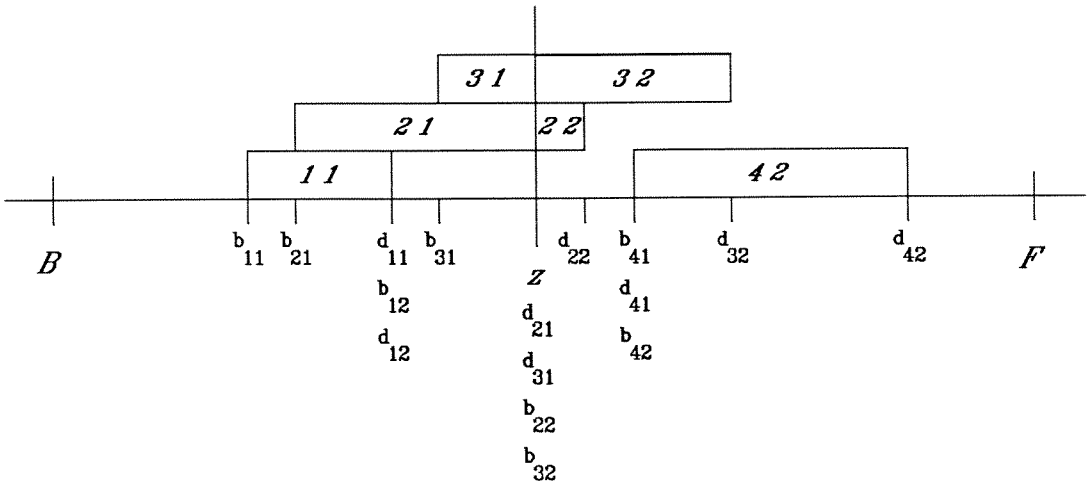
$$F - z = \sum\nolimits_i p_{i2} . \tag{5}$$

Figure 2: The Definitions of $g(t)$ and z.



The shaded area is g(z) .

Figure 3: The Definition of LBP .



Because our original problem is assumed to have one cluster, LBQ has exactly two clusters, $\mathcal{M}_1 = \{i1 : i \in \mathcal{J}\}$ and $\mathcal{M}_2 = \{i2 : i \in \mathcal{J}\}$. By (4) and (5), in an optimal solution to LBQ the jobs in $\mathcal{M}_1$ (resp., $\mathcal{M}_2$) will be scheduled in the time interval $[B, z]$ (resp., $[z,F]$). We will show that no jobs in $\mathcal{M}_1$ are late and no jobs in $\mathcal{M}_2$ are early. This fact will make the optimal cost for LBQ easy to calculate.

Corollary 7. If $B \le t \le z$ then $\Sigma_{i:d_i \le t} p_i \le t - B$. If $F \ge t \ge z$ then $\Sigma_{i:b_i < t} p_i \ge t - B$ and $\Sigma_{i:b_i \ge t} p_i \le F - t$.

Proof. The first assertion follows from the definition of z. Assume $F \ge t \ge z$. If $t \ge max_i\, d_i$ then the result holds. If $z < x \le t < max_i\, d_i$ then since there is only one cluster and since $\min_i b_i \le z$, there must be a job j such that $b_j < x < d_j$. This implies that $g'(x) \ge 1$ for all $z < x \le t$. Thus $t \le z + \int_z^t g'(\xi)\, d\xi = g(t) - g(z) + z \le [\Sigma_{i:b_i<t} p_i] - [z - B] + z = \Sigma_{i:b_i<t} p_i + B$. In addition $F - t \ge F - [B + \Sigma_{i:b_i<t} p_i] = \Sigma_{i:b_i \ge t} p_i$. □

Lemma 8. If Q has a single cluster then $LB \equiv \frac{1}{2}(\Sigma_i p_{i2})^2 + \frac{1}{2}\Sigma_i p_{i2}^2 + \Sigma_i p_{i2}(z - d_i)$ is a lower bound on the cost of an optimal solution to Q.

Proof. Let $S = \{1, 2, ..., J\}$ be a schedule for Q, and let $S^*(t) = \{11, 12, 21, 22, 31, ..., J2\}$ be the corresponding schedule for LBQ. We claim that $p_i(c_i(S) - d_i)^+ = p_{i1}(c_{i1}(S^*) - d_{i1})^+ + p_{i2}(c_{i2}(S^*) - d_{i2})^+$. If either $p_{i1} = 0$ or $p_{i2} = 0$ then the claim clearly holds. If $p_{i1} > 0$ and $p_{i2} > 0$ then $b_i < z = d_{i1} < d_i$, so $c_i(S) - d_i = c_{i1}(S^*) - d_{i1} + c_{i2}(S^*) - z$. Therefore our claim holds, so the cost of S and the cost of S^* are equal. Futhermore the cost of an optimal solution to LBQ is a lower bound on the cost of an optimal solution to Q.

Let S^* be the schedule for LBQ in which the jobs $\{i1 : i \in J\}$ in EDD sequence are followed by the jobs $\{i2 : i \in J\}$ in EBD sequence. We will show that S^* is optimal for LBQ. By Lemma 6, all jobs of the form $i1$ precede all jobs of the form $j2$ in an optimal schedule for LBQ. Equation (4) implies that the last job of the form $i1$ completes at time z. Since $d_{i1} = min(d_i, z)$, Corollary 7 implies that none of these jobs is late in S^*. By Corollary 5, the EDD sequence that S^* uses is optimal for these jobs. Jobs of the form $i2$ are sequenced by EBD in S^*, starting at time z. Since $b_{i2} = max(z, b_i)$, by Corollary 7 none of these jobs is early. Corollaries 4 and 5 imply that the EBD sequence is optimal for these jobs, so S^* is optimal for LBQ. All that remains is to compute the cost of this schedule.

Let the jobs $i \in J$ be indexed in increasing order of b_{i2}, i.e., $b_{12} \le b_{22} \le \ldots \le b_{J2}$. Then the cost of S^* is

$$LB = \sum_i p_{i2}(c_{i2}(S^*) - d_i) = \sum_i p_{i2}\left[z + \sum_{j=1}^{i} p_{j2} - d_i \right]$$

$$- \frac{1}{2}\left[\sum_i p_{i2} \right]^2 + \frac{1}{2}\sum_i p_{i2}^2 + \sum_i p_{i2}(z - d_i) . \quad \square$$

The lower bound has another useful form. Let $U = \{i : d_i \le z\}$, $V = \{i : b_i \le z < d_i\}$, and $W = \{i : z < b_i\}$. Let $P_W = \Sigma_{i \in W} p_i$. If $i \in U$ then $p_{i2} = 0$, if $i \in V$ then $p_{i2} = d_i - z$, and if $i \in W$ then $p_{i2} = p_2$. As in the proof of Lemma 8, we assume that the jobs are indexed in increasing order of b_{i2}. Therefore if $i \in U$, $j \in V$, and $k \in W$ then $i < j < k$. Since $\Sigma_i p_{i2} = F - z$, we have

$$
\begin{aligned}
LB &= \sum\nolimits_j p_{j2}\Big[z + \sum\nolimits_{i=1}^{j} p_{i2} - d_j\Big] \\
&= 0 + \sum\nolimits_{j\in V} p_{j2}\Big[\sum\nolimits_{i\in V: i<j} p_{i2}\Big] + \sum\nolimits_{j\in W} p_j\Big[F - \sum\nolimits_{i>j} p_i - d_j\Big] \\
&= \Big\{\tfrac{1}{2}\Big[\sum\nolimits_{j\in V} p_{j2}\Big]^2 - \tfrac{1}{2}\sum\nolimits_{j\in V} p_{j2}^2\Big\} + \\
&\quad \Big\{-\tfrac{1}{2}p_W^2 + \tfrac{1}{2}\sum\nolimits_{j\in W} p_j^2 + \sum\nolimits_{j\in W} p_j(F - d_j)\Big\}.
\end{aligned} \tag{6}
$$

Note that (6) would still hold if we had defined $U = \{i : d_i < z\}$, $V = \{i : b_i < z \le d_i\}$, and $W = \{i : z \le b_i\}$.

SECTION 3. WORST–CASE PERFORMANCE OF EARLIEST-GAMMA-DATE SCHEDULES.

In this section we evaluate the worst-case performance of EγD heuristics. Recall that $0 \le \theta \le 1$ and that $\gamma_i \equiv \theta b_i + \bar{\theta} d_i$. The E$\gamma$D heuristics sequence the jobs in increasing order of γ_j. For heuristics in this class, we will show that

$$
R \equiv [C(\text{E}\gamma\text{D}) - LB]/[\textstyle\sum_i p_i^2] \le \tfrac{1}{2}(\theta \vee \bar{\theta})^2. \tag{7}
$$

The EγD heuristics differ only in the tie–breaking rules that they use. The following lemma implies that from the point of view of determining how large R can get, any EγD heuristic can be used.

LEMMA 9. All EγD heuristics with the same θ have the same worst-case value of R, independent of which tie-breaking rule is employed.

PROOF. For any EγD sequence S and any $\epsilon > 0$, the due dates (or equivalently, the γ_i's) can be perturbed by an amount less than ϵ so that after perturbation S is the only EγD sequence. Since the cost of the schedule and of the lower bound are continuous functions of the due dates, the result follows. □

The examples below illustrate that the bound of Equation (7) is tight.

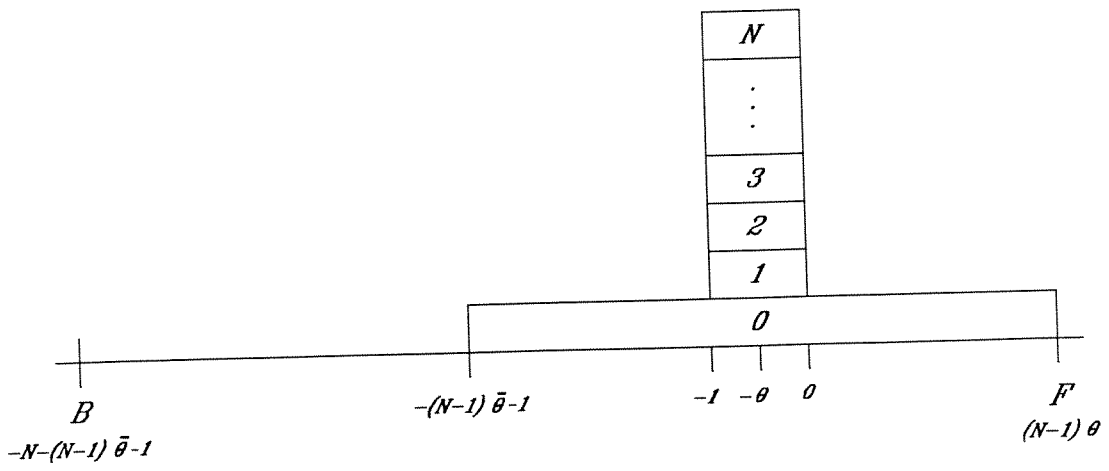
EXAMPLE 1.

Data

$J = \{0, 1, ..., N\}$.

$B = -N - (N-1)\bar{\theta} - 1$, $F = (N-1)\theta$

i	p_i	γ_i	b_i	d_i
0	N	$-\theta$	$-(N-1)\bar{\theta} - 1$	$(N-1)\theta$
i , $1 \leq i \leq N$	1	$-\theta$	-1	0



Schedules

$S = \{0, 1, ..., N\}$.

The following algebraic identity is useful. If $0 < \nu < N$ and $\delta = \lceil \nu \rceil - \nu$ then

$$\sum_{i=1}^{N} (i-\nu)^+ = \sum_{i=\nu+\delta}^{N} [(i-\nu-\delta)+\delta]$$

$$= \frac{1}{2}(N-\nu-\delta)(N-\nu-\delta+1) + \delta(N-\nu-\delta+1)$$

$$= \frac{1}{2}(N-\nu)(N-\nu+1) + \frac{1}{2}\delta(1-\delta) .$$

Note that $0 \leq \delta < 1$ by definition. Using this identity with $\nu = (N-1)\bar{\theta} + 1$, we have

$$C(S) = \sum_{i=1}^{N} (i-(N-1)\bar{\theta}-1)^+ = \frac{1}{2}(N\theta-\bar{\theta})(N\theta-\bar{\theta}-1) + \frac{1}{2}\delta(1-\delta)$$

$$= \frac{\theta^2}{2} N^2 + O(N) .$$

$\bar{S} = \{1, ..., N, 0\}$.

$C(\bar{S}) = 0$.

The Cost Ratio is

$$\frac{C(S) - C(\bar{S})}{\Sigma_j p_j^2} \geq \frac{\theta^2 N^2/2 + O(N)}{N^2 + N} = \frac{\theta^2}{2} + O(\frac{1}{N}) .$$

EXAMPLE 2.

The second example is the reversal in time of the Example 1, in the sense of Corollary 4. Because time is reversed, the roles of θ and $\bar{\theta}$ are interchanged, so a cost ratio of $\bar{\theta}^2/2 + O(1/N)$ is obtained. The details of the example are omitted.

Together, these two examples show that the bound in (7) cannot be improved as long as the right-hand side is a function of θ alone. The remainder of this section is dedicated to showing that (7) holds.

By Lemma 9 all $E\gamma D$ heuristics have the same worst–case value of R. Therefore we choose to study a particular $E\gamma D$ schedule that is easy to analyze. This is the schedule $E\gamma/B\gamma p$ (earliest gamma, biggest $(\gamma_j - z)p_j$), in which ties are broken by putting the job with largest $(\gamma_j - z)p_j$ first. Thus ties are broken by the shortest processing time rule if $\gamma_j < z$ and by the longest processing time rule if $\gamma_j > z$. In the sequel we use $E\gamma/B\gamma p$ to refer either to the schedule or to the heuristic.

In order to show that (7) holds we first consider the problem of selecting due dates so as to maximize R.

Lemma 10. Assume that there is only one cluster. For the heuristic $E\gamma/B\gamma p$, R is maximized when $\gamma_j = \gamma$ for all j.

Proof. Suppose that for a fixed set of processing times, the current set of due dates lexicographically maximizes

$$\left[R\,,\ -\sum\nolimits_{i<j}(\gamma_j - \gamma_i)^2\right]. \tag{8}$$

Let the jobs be indexed so that $E\gamma/B\gamma p = \{1, 2, .., J\}$, and let i be the last job in $E\gamma/B\gamma p$ which has $\gamma_i = \gamma_1$. Let j be the first job in $E\gamma/B\gamma p$ that has $\gamma_j = \gamma_J$. Assume by way of contradiction that the lemma fails, i.e., $\gamma_j > \gamma_i$. Note that because of our choice of i and j, if we perturb the problem data by slightly increasing d_i or by slightly decreasing d_j, without changing p_i and p_j, then the $E\gamma/B\gamma p$ schedule does not change, and we will still have only one cluster. However such a perturbation will cause a strict increase in $-\Sigma_{i<j}(\gamma_j - \gamma_i)^2$. In each of the following six cases, we will find a

perturbation of this type that contradicts the minimality of (8). Recall that $LB = \frac{1}{2}(\Sigma_i p_{i2})^2 + \frac{1}{2}\Sigma_i p_{i2}^2 + \Sigma_i p_{i2}(z - d_i)$.

Case 1: $d_i < z$. Set $d_i \leftarrow d_i + \alpha$, α small and positive. LB is unchanged. The definition of i implies that for all $k \leq i$, $p_k \leq p_i$ and $\gamma_k = \gamma_i$, so $d_k \leq d_i < z$. Corollary 7 implies that i is not late, so $C(\mathrm{E}\gamma/\mathrm{B}\gamma p)$ does not change. This contradicts the minimality of (8).

Case 2: $b_j > z$. Set $d_j \leftarrow d_j - \alpha$, α small and positive. LB goes up by αp_j. The definition of j implies that for all $k \geq j$, $p_k \leq p_j$ and $\gamma_k = \gamma_j$, so $b_k \geq b_j$. Corollary 7 implies that j is not early, so $C(\mathrm{E}\gamma/\mathrm{B}\gamma p)$ goes up by αp_j. This contradicts the minimality of (8).

Case 3: $b_i \geq z$. Set $b_i \leftarrow b_i + \alpha$, α small and positive. Then LB goes down by αp_i. $C(\mathrm{E}\gamma/\mathrm{B}\gamma p)$ cannot go down by more than αp_i, so we have contradicted the minimality of (8).

Case 4: $d_j \leq z$. Set $d_j \leftarrow d_j - \alpha$, α small and positive. Then LB is unchanged. $C(\mathrm{E}\gamma/\mathrm{B}\gamma p)$ cannot decrease, so we have contradicted the minimality of (8).

Case 5: $b_i < z \leq d_i$, $b_j \leq z < d_j$, i *is early in* $\mathrm{E}\gamma/\mathrm{B}\gamma p$. Let N be the number of jobs $k \neq i$ such that $b_k \leq z < d_k$. We increase d_i by αN, α small and positive. Let z^0 and z^α be the initial and final values of z, respectively, and let $g_0(x)$, $g_\alpha(x)$, LB^0, and LB^α be similarly defined. Note that $g_\alpha(z^0) = g_0(z^0) - \alpha N$ and $g'_\alpha(z^0) = N + 1$. Therefore $g_\alpha(z^0 + \alpha) = g_\alpha(z^0) + \alpha \times g'_\alpha(z^0) = g_0(z^0) + \alpha = z^0 + \alpha - B$, so $z^\alpha = z^0 + \alpha$.

Since i is early, $C(\mathrm{E}\gamma/\mathrm{B}\gamma p)$ is constant. Let $S \equiv \{k \neq i: b_k \leq z < d_k\}$. Using (6),

$$LB^0 = \frac{1}{2}\left[p_{i2} + \sum_{k \in S} p_{k2} \right]^2 - \frac{1}{2}\left[p_{i2}^2 + \sum_{k \in S} p_{k2}^2 \right] + A$$

where A is the second term in braces in (6).

$$LB^{\alpha} = \frac{1}{2}\left[(p_{i2} + (N\text{-}1)\alpha) + \sum_{k\in S} (p_{k2} - \alpha) \right]^2$$

$$- \frac{1}{2}\left[(p_{i2} + (N\text{-}1)\alpha)^2 + \sum_{k\in S} (p_{k2} - \alpha)^2 \right] + A .$$

Noting that $|S| = N$, we have

$$LB^{\alpha} - LB^{0} = \alpha\left[-p_{i2} - \sum_{k\in S} p_{k2} - (N\text{-}1)p_{i2} + \sum_{k\in S} p_{k2} \right] + \frac{\alpha^2}{2}\left[1 - (N\text{-}1)^2 - N \right]$$

$$= -p_{i2}N\alpha - \frac{N(N\text{-}1)}{2}\alpha^2 < 0 .$$

This contradicts the minimality of (8).

Case 6: $b_i < z \le d_i$, $b_j \le z < d_j$, i *is not early in* $\mathrm{E}\gamma/\mathrm{B}\gamma p$. This case is reminiscent of Case 5. Since $b_j \le z \le d_i \le c_i(\mathrm{E}\gamma/\mathrm{B}\gamma p) < c_j(\mathrm{E}\gamma/\mathrm{B}\gamma p)$, job j is late in $\mathrm{E}\gamma/\mathrm{B}\gamma p$. Let N be the number of jobs $k \neq j$ such that $b_k < z^0 \le d_k$. We decrease d_j by αN, α small and positive. Let z^0 and z^{α} be the initial and final values of z, respectively, and let $g_0(x)$, $g_{\alpha}(x)$, LB^0, LB^{α}, $C^0(\mathrm{E}\gamma/\mathrm{B}\gamma p)$, and $C^{\alpha}(\mathrm{E}\gamma/\mathrm{B}\gamma p)$ be similarly defined. Note that $g_{\alpha}(z^0) = g_0(z^0) + \alpha N$ and $g'_{\alpha}(z^0) = N + 1$. Therefore $g_{\alpha}(z^0 - \alpha) = g_{\alpha}(z^0) - \alpha \times g'_{\alpha}(z^0) = g_0(z^0) - \alpha = z^0 - \alpha - B$, so $z^{\alpha} = z^0 - \alpha$.

$C^{\alpha}(\mathrm{E}\gamma/\mathrm{B}\gamma p) - C^0(\mathrm{E}\gamma/\mathrm{B}\gamma p) = p_j N\alpha$. Let $S \equiv \{k \neq j : b_k < z \le d_k\}$. Using (6) (with V defined by $V = \{k : b_k < z \le d_k\}$),

$$LB^0 = \frac{1}{2}\left[p_{j2} + \sum_{k\in S} p_{k2} \right]^2 - \frac{1}{2}\left[p_{j2}^2 + \sum_{k\in S} p_{k2}^2 \right] + A$$

where A is the second term in braces in (6).

$$LB^{\alpha} = \frac{1}{2}\left[(p_{j2} - (N\text{-}1)\alpha) + \sum_{k\in S} (p_{k2} + \alpha) \right]^2$$

$$- \frac{1}{2}\left[(p_{j2} - (N\text{-}1)\alpha)^2 + \sum_{k\in S} (p_{k2} + \alpha)^2 \right] + A .$$

Noting that $|S| = N$ we have

$$[C^0(\mathrm{E}\gamma/\mathrm{B}\gamma p) - LB^0] - [C^{\alpha}(\mathrm{E}\gamma/\mathrm{B}\gamma p) - LB^{\alpha}]$$

$$= \alpha\left[-p_j N + p_{j2} + \sum_{k\in S} p_{k2} + (N\text{-}1)p_{j2} - \sum_{k\in S} p_{k2} \right] + \frac{\alpha^2}{2}\left[1 - (N\text{-}1)^2 - N \right]$$

$$= -p_{j1} N\alpha - \frac{N(N\text{-}1)}{2}\alpha^2 < 0 .$$

This contradicts the minimality of (8). □

Lemma 10 implies that in searching for an upper bound on R for a single cluster, we can confine ourselves to data sets with $\gamma_i = \gamma$ for all i. Note that in Example 1, all jobs have the same value of γ_i.

Lemma 11. If there is only one cluster and $\gamma_i = \gamma$ for all i then $R \le \frac{1}{2}(\theta \vee \bar{\theta})^2$.

Proof. The lower bound for an equal–gamma data set is

$$LB = \frac{1}{2}\left[\sum_i p_{i2}\right]^2 + \frac{1}{2}\sum_i p_{i2}^2 + \sum_i p_{i2}(z - d_i)$$

$$= \frac{1}{2}(F - z)^2 + \frac{1}{2}\sum_i p_{i2}^2 + \sum_i p_{i2}(z - \gamma - \theta p_i)$$

$$= \frac{1}{2}(F - z)^2 + (z - \gamma)(F - z) + \frac{1}{2}\sum_i p_{i2}^2 - \theta\sum_i p_{i2} p_i .$$

Assume that the $\mathrm{E}\gamma/\mathrm{B}\gamma\mathrm{p}$ sequence is $\{1, 2, \ldots, J\}$. Let k be the job that straddles γ in $\mathrm{E}\gamma/\mathrm{B}\gamma p$, and let the completion time of k in $\mathrm{E}\gamma/\mathrm{B}\gamma p$ be $\gamma + \bar{\rho}p_k$. Then k and ρ are given by

$$\gamma - B = \sum_{i<k} p_i + \rho p_k, \quad 0 \le \rho \le 1.$$

$$F - \gamma = \sum_{i>k} p_i + \bar{\rho} p_k \quad \text{where } \bar{\rho} = 1 - \rho.$$

Since k straddles γ, k is either the first late job or the last early job. The cost of $\mathrm{E}\gamma/\mathrm{B}\gamma p$ is

$$C(\mathrm{E}\gamma/\mathrm{B}\gamma p) = \sum_{i \ge k} p_i[(c_i(\mathrm{E}\gamma/\mathrm{B}\gamma p) - \gamma) - (d_i - \gamma)]^+$$

$$= p_k[\bar{\rho}p_k - \theta p_k]^+ + \sum_{i>k} p_i \left[\left[\bar{\rho}p_k + \sum_{j:k<j\le i} p_j\right] - \theta p_i\right]$$

$$= p_k^2(\bar{\rho} - \theta)^+ + \frac{1}{2}\left[\sum_{i>k} p_i\right]^2 + \frac{1}{2}\sum_{i>k} p_i^2 + \left[\sum_{i>k} p_i\right]\bar{\rho}p_k - \left[\sum_{i>k} p_i^2\right]\theta.$$

Completing the square on the second and fourth terms,

$$C(\mathrm{E}\gamma/\mathrm{B}\gamma p) = \frac{1}{2}\left[\sum_{i>k} p_i + \bar{\rho}p_k\right]^2 + p_k^2[(\bar{\rho} - \theta)^+ - \tfrac{1}{2}\bar{\rho}^2] + \left[\tfrac{1}{2} - \theta\right]\left[\sum_{i>k} p_i^2\right]$$

$$= \frac{1}{2}(F - \gamma)^2 + p_k^2[(\bar{\rho} - \theta)^+ - \tfrac{1}{2}\bar{\rho}^2] + \left[\tfrac{1}{2} - \theta\right]\left[\sum_{i>k} p_i^2\right].$$

Therefore

$$C(\mathrm{E}\gamma/\mathrm{B}\gamma p) - LB = \left\{ \tfrac{1}{2}(F-\gamma)^2 + p_k^2[(\bar{\rho}-\theta)^+ - \tfrac{1}{2}\bar{\rho}^2] + \left[\tfrac{1}{2}-\theta\right]\left[\sum_{i>k} p_i^2\right] \right\}$$

$$- \left\{ \tfrac{1}{2}(F-z)^2 + (z-\gamma)(F-z) + \tfrac{1}{2}\sum_i p_{i2}^2 - \theta \sum_i p_{i2}p_i \right\}$$

$$= \tfrac{1}{2}(z-\gamma)^2 + p_k^2[(\bar{\rho}-\theta)^+ - \tfrac{1}{2}\bar{\rho}^2] + \left[\tfrac{1}{2}-\theta\right]\left[\sum_{i>k} p_i^2\right]$$

$$- \tfrac{1}{2}\sum_i p_{i2}^2 + \theta \sum_i p_{i2}p_i$$

$$= \tfrac{1}{2}(z-\gamma)^2 + p_k^2[(\bar{\rho}-\theta)^+ - \tfrac{1}{2}\bar{\rho}^2] + \left[\tfrac{1}{2}-\theta\right]\left[\sum_{i>k} p_i^2\right]$$

$$- \tfrac{1}{2}\sum_i [p_{i2}-\theta p_i]^2 + \tfrac{1}{2}\theta^2 \sum_i p_i^2 .$$

We will show that $R \le \frac{1}{2}(\theta \vee \bar{\theta})^2$. There are two cases.

Case 1: $\theta \ge 1/2$. In this case it suffices to show that $0 \ge C(\mathrm{E}\gamma/\mathrm{B}\gamma p) - LB - \frac{1}{2}\theta^2 \Sigma_i p_i^2$. Note that in this case $[(\bar{\rho}-\theta)^+ - \frac{1}{2}\bar{\rho}^2] \le 0$.

$$C(\mathrm{E}\gamma/\mathrm{B}\gamma p) - LB - \tfrac{1}{2}\theta^2 \sum_i p_i^2 =$$

$$\tfrac{1}{2}(z-\gamma)^2 + p_k^2[(\bar{\rho}-\theta)^+ - \tfrac{1}{2}\bar{\rho}^2] + \left[\tfrac{1}{2}-\theta\right]\left[\sum_{i>k} p_i^2\right] - \tfrac{1}{2}\sum_i [p_{i2}-\theta p_i]^2$$

$$\le \tfrac{1}{2}(z-\gamma)^2 - \tfrac{1}{2}\sum_i (p_{i2}-\theta p_i)^2 .$$

Since $min_i\, b_i \le z \le max_i\, d_i$, and we have a single cluster, there is at least one job i such that $b_i \le z \le d_i$. For that job, $p_{i2} - \theta p_i = \gamma - z$, so $(p_{i2}-\theta p_i)^2 = (z-\gamma)^2$. The result follows.

Case 2: $\theta \le 1/2$. In this case, it suffices to show that $0 \ge C(\mathrm{E}\gamma/\mathrm{B}\gamma p) - LB - \frac{1}{2}\bar{\theta}^2 \Sigma_i p_i^2$. Note that in this case $[(\bar{\rho}-\theta)^+ - \frac{1}{2}\bar{\rho}^2] \le \frac{1}{2} - \theta$.

$$C(\mathrm{E}\gamma/\mathrm{B}\gamma p) - LB - \tfrac{1}{2}\bar{\theta}^2 \sum_i p_i^2$$

$$= \tfrac{1}{2}(z-\gamma)^2 + p_k^2[(\bar{\rho}-\theta)^+ - \tfrac{1}{2}\bar{\rho}^2] +$$

$$\left[\tfrac{1}{2}-\theta\right]\left[\sum_{i>k} p_i^2\right] - \tfrac{1}{2}\sum_i [p_{i2}-\theta p_i]^2 + \tfrac{1}{2}(\theta^2-\bar{\theta}^2)\sum_i p_i^2$$

$$\le \tfrac{1}{2}(z-\gamma)^2 + \left[\tfrac{1}{2}-\theta\right]\left[p_k^2 + \sum_{i>k} p_i^2\right] - \tfrac{1}{2}\sum_i [p_{i2}-\theta p_i]^2 + \left[\theta-\tfrac{1}{2}\right]\sum_i p_i^2$$

$$\le \tfrac{1}{2}(z-\gamma)^2 - \tfrac{1}{2}\sum_i [p_{i2}-\theta p_i]^2 \le 0,$$

where the last inequality follows from the argument in Case 1. □

We are now finally ready to prove Theorem 1.

Theorem 1. The cost of an EγD schedule can not exceed the cost of an optimal schedule by more than $\frac{1}{2}(\theta \vee \bar{\theta})^2 \Sigma_i p_i^2$ where $\gamma_i = b_i + \bar{\theta} p_i$.

Proof. Lemmas 8, 9, 10, and 11 imply the result if there is only one cluster, assuming that (1) holds. However transforming Q so as to enforce (1) has no effect on R, so the result holds for all one-cluster problems. If there is more than one cluster then we decompose the problem by cluster. Lemma 6 implies that the sum of the lower bounds for the individual clusters constitutes a lower bound on the cost of any solution to the original problem. Clearly the cost of an EγD schedule for the original problem is a sum of costs of EγD schedules for each of the the one-cluster subproblems. The same applies to the

denominator of R. The result follows by applying Lemmas 10 and 11 to each of the sub-problems and by noting that $x_k \geq 0$, $y_k > 0$, and $x_k/y_k \leq X$ for all k implies that $[\Sigma_k x_k]/[\Sigma_k x_k] \leq X$. □

SECTION 4. PARALLEL IDENTICAL MACHINES.

In this section we obtain bounds on the costs of EγD heuristics for systems with parallel, identical machines. An EγD schedule for parallel machines is obtained by applying the following rule: Whenever one of a bank of parallel machines becomes available, we schedule the available job with earliest gamma-date on it.

Throughout this section we assume that all of the machines become available for processing at time $B = 0$. Let Q_2 be the parallel-machine scheduling problem that is identical to Q, except that there are M parallel, identical machines instead of one. Let Q_1 be the problem of scheduling the same jobs on a single machine that works M times as fast as any of the machines of Q_2. We will show that an optimal solution to Q_1 is a lower bound on the cost of an optimal solution to Q_2, and we will use this lower bound to obtain results for EγD schedules that are similar to the bounds obtained in Section 4.

LEMMA 12. The cost of an optimal solution to Q_1 is a lower bound on the cost of an optimal solution to Q_2.

PROOF. Let S_2 be a schedule for Q_2, and let S_1 be a schedule for Q_1 that is derived from S_2 as follows. If c_i^2 is the completion time of job i in S_2 and $c_i^2 < c_j^2$, then i must precede j in S_1. Let c_i^1 be the completion time of job i in S_1. Then

$$c_j^1 \leq \sum_{i: c_i^2 \leq c_j^2} \frac{p_i}{M} \leq c_j^2.$$

Therefore $C(S_2) \geq C(S_1)$. □

Let $E\gamma D_2$ be the earliest-gamma-date schedule for Q_2 and let $E\gamma D_1$ be the earliest-gamma-date schedule for Q_1. We assume that both of these schedules use the same tie-breaking rule. The next lemma relates the costs of these two schedules.

LEMMA 13. $C(E\gamma D_2) \leq C(E\gamma D_1) + \frac{M-1}{M} \Sigma_i p_i^2$.

PROOF. Let c_i^1 be the completion time of job i in $E\gamma D_1$, and let c_i^2 be the completion time of job i in $E\gamma D_2$. Let $E\gamma D_1 = \{1, 2, ..., J\}$, so that $c_j^1 = \frac{1}{M} \Sigma_{i \leq j} p_i$. When job j is scheduled in Q_2 it is put on the first available machine, so $c_j^2 - p_j \leq \frac{1}{M} \Sigma_{i<j} p_i$. Therefore $c_j^2 - c_j^1 \leq \frac{M-1}{M} p_j$, and $C(E\gamma D_2) - C(E\gamma D_1) \leq \frac{M-1}{M} \Sigma_i p_i^2$. □

Let Opt be an optimal schedule for Q_2.

THEOREM 14. $\frac{C(E\gamma D_2) - C(Opt)}{\Sigma_i p_i^2} \leq \frac{1}{2M} (\theta \vee \bar{\theta})^2 + \frac{M-1}{M}$.

PROOF. Since Q_1 is a single–machine scheduling problem of the type studied in sections 2 through 4, the lower bound derived in section 3 applies to it. Let LB_1 be the value of this lower bound. Note that instead of having the weights equal to the prcessing times, in Q_1 the weight of job i is p_i and the processing time of job i is p_i/M. Therefore the value of the lower bound is $LB_1 = \frac{1}{M} \Sigma_i p_i^2$. Lemmas 12 and 13 and Theorem 1 imply that $C(E\gamma D_2) - C(Opt) \leq [C(E\gamma D_1) + \frac{M-1}{M} \Sigma_i p_i^2] - LB_1 \leq \frac{M-1}{M} \Sigma_i p_i^2 + \frac{1}{2M} (\theta \vee \bar{\theta})^2 \Sigma_i p_i^2$.

□

We conjecture that the bound of Theorem 14 is not tight. The following example shows that this bound can not be reduced below $\bar{\theta}^2/2 + (M-1)/2M$.

EXAMPLE 3.

Problem Data

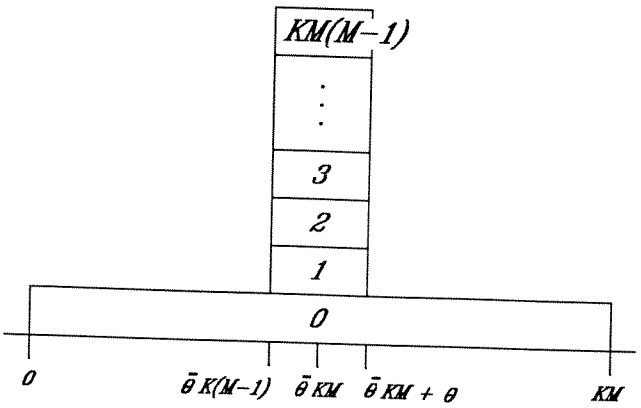
M = the number of machines.

K a large integer.

$\mathcal{J} = \{0, 1, ..., KM(M-1)\}$: The set of jobs.

$p_0 = KM$; $p_i = 1$ for all $i \geq 1$.

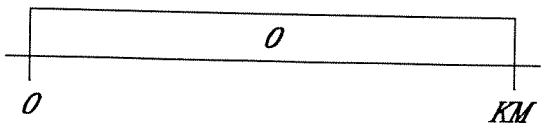
$\gamma_i = \bar{\theta} KM$ for all jobs i .



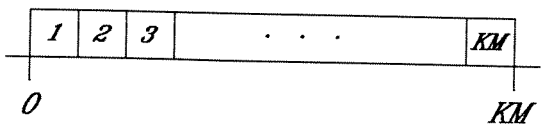
Schedules

Schedule S^* : On machine 1 schedule job 0 . On machine $m > 1$ schedule jobs i , $(m-2)KM < i \leq (m-1)KM$.

Machine 1 .



Machine 2 .



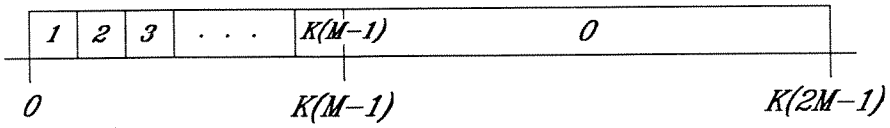
Using the algebraic identity derived as part of Example 1 with $\nu = \bar{\theta}KM + \theta$, we have

$$C(S^*) = (M\text{–}1) \sum_{i=1}^{KM} (i - \bar{\theta}KM - \theta)^+$$

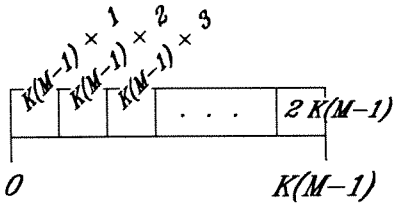
$$= (M\text{–}1)\,\tfrac{1}{2}\,(KM - \bar{\theta}KM - \bar{\theta})\,(KM + 1 - \theta KM - \bar{\theta}) \;=\; \tfrac{\theta^2}{2}\,K^2M^2(M\text{–}1) + O(KM^2)\,.$$

Schedule S: On machine 1 schedule jobs $1, 2, \ldots, K(M\text{–}1), 0$. On machine $m > 1$ schedule jobs i, $(m\text{–}1)K(M\text{–}1) < i \leq mK(M\text{–}1)$.

Machine 1.



Machine 2.



Using the algebraic identity of Example 1 with $\nu = \bar{\theta}KM + \theta$, we have

$$C(S) \;=\; (KM)(K(M\text{–}1)) \;+\; M \sum_{i=1}^{K(M\text{–}1)} (i - \bar{\theta}KM - \theta)^+$$

$$\geq\; K^2M(M\text{–}1) \;+\; M\,\tfrac{1}{2}\,[K(M-1) - \bar{\theta}KM - \theta + 1][K(M-1) - \bar{\theta}KM - \theta]$$

$$= K^2\left(\tfrac{\theta^2}{2}\,M^3 + \bar{\theta}M^2 - \tfrac{1}{2}\,M\right) \;+\; O(KM^2)\,.$$

$$\sum_i p_i^2 = K^2M^2 + KM(M-1) .$$

$$\frac{C(S) - C(S^*)}{\Sigma_i p_i^2} \geq \frac{K^2[M^2(\bar{\theta} + \theta^2/2) - M/2] + O(KM^2)}{K^2M^2 + O(KM^2)}$$, which tends to $\frac{\bar{\theta}^2}{2} + \frac{M-1}{2M}$

as K tends to infinity.

SECTION 5: COMPUTATIONAL TESTS.

Computational tests were performed for single-machine problems. The main goal of these tests was to determine how sensitive the performance of EγD heuristics is to the assumption that the weights are proportional to the processing times.

The problem data were generated as follows. The number of jobs J was set at 5, 10, or 15. The processing times were drawn from a discrete uniform distribution from 1 to 10. The weights were generated by assuming that for job i, $log_2(w_i/p_i)$ is uniformly distributed on the interval $(-\alpha, \alpha)$ where $\alpha = 0, 0.25, 0.5, 0.75, 1$. The begin dates were generated according to a uniform distribution from $U - V$ to $U + V$. Note that $5.5J$ is the expected total processing time. $U/(5.5J)$ was selected from 0.25, 0.29, 0.35, 0.42, and 0.5. V was selected from 0, $U/2$, and U. U is a measure of how tight the due dates are, and V is a measure of how spread out they are.

For each combination of J, α, $U/(5.5J)$, and V we randomly generated ten problem instances, making 2250 problems in all. Each problem was solved to optimality using a standard dynamic programming algorithm. EγD schedules were computed for $\gamma_i = d_i$ (EDD), $\gamma_i = b_i$ (EBD), and $\gamma_i = (d_i + b_i)/2$ (EHD). For each value of γ the ratio

$$Q = \frac{C(\text{E}\gamma\text{D}) - C(Opt)}{\Sigma_i p_i^2}$$

was computed, where *Opt* is the optimal schedule. The results appear in Table 1 below.

Table 1: Average Values of Q.

Sensitivity to the Disproportionality of the Weights.

		EBD	EDD	EHD
alpha	0	0.027	0.004	0.002
	0.25	0.089	0.067	0.064
	0.50	0.165	0.144	0.141
	0.75	0.255	0.225	0.223
	1.00	0.325	0.290	0.290

Sensitivity to the Number of Jobs and the Weights: The EHD Rule.

	J	5	10	15
	0	0.002	0.001	0.002
alpha	0.25	0.022	0.061	0.107
	0.50	0.048	0.139	0.237
	0.75	0.062	0.214	0.392
	1.00	0.109	0.294	0.484

Sensitivity to Due Date Tightness and Spread: The EHD Rule.

	V/U	0.0	0.5	1.0
	0.25	0.242	0.239	0.236
$U/5.5J$	0.29	0.211	0.189	0.165
	0.35	0.190	0.144	0.103
	0.42	0.145	0.079	0.063
	0.5	0.105	0.048	0.017

The EHD and EDD heuristics perform slightly better than the EBD heuristic, and for $\alpha = 0$ the EHD heuristic out-performs the EDD heuristic. But all of the heuristics exhibit comparable performance. The sensitivity of all three heuristics to the problem parameters is also similar. The EγD heuristics do worse as the variability in w_i/p_i increases, but they perform extremely well when the variations in w_i/p_i are as large as 20% above or below the mean. The EγD heuristics are insensitive to problem size if the weights are proportional to the mean ($\alpha = 0$), but they deteriorate with problem size if α is positive. The algorithms work well when U and V are large, i.e., when the due dates are evenly spread out over the time horizon so that not many jobs are seriously late.

We also computed $[C(\text{E}\gamma\text{D}) - C(Opt)]/C(Opt)$ for these problems, but the results were not meaningful. $C(Opt)$ can be very small, making the ratio very large even though $C(\text{E}\gamma\text{D})$ is quite close to $C(Opt)$. Furthermore the large ratios have a disproportionate and capricious effect on averages of the type reported above.

SECTION 6: A PSEUDO-POLYNOMIAL TIME ALGORITHM.

In this section we present some important properties of optimal schedules for 1–machine weighted-tardiness scheduling problems with ptoportional weights, and we give a pseudo–polynomial time algorithm for solving Q. Recall that $\mathcal{E} = \{i : c_i(S) \leq d_i\}$, the

set of jobs that are either early or exactly on time in schedule S, and $\mathcal{L} = \{i : c_i(S) > d_i\}$, the set of (strictly) late jobs in schedule S. Note that by convention, a job which is exactly on time is considered to be early.

LEMMA 15. If the scheduling problem Q has only one cluster then in a schedule which lexicographically minimizes $(C(S), |\mathcal{E}|)$, the jobs in $\mathcal{E}$ precede the jobs in $\mathcal{L}$.

PROOF. Suppose that schedule S does not satisfy the conditions of Lemma 15 Then there is a pair of jobs j,k satisfying j directly precedes k, $j \in \mathcal{L}$, and $k \in \mathcal{E}$. Without loss of generality let the completion time of job j be time zero. Then $b_j < d_j < 0 \leq b_k < d_k$. Since Q has only one cluster, there is a job ℓ satisfying $b_\ell < 0 < d_\ell$. Let $c_\ell(S)$ be the completion time of job ℓ. There are two cases to consider.

Case 1: $c_\ell(S) > 0$. Let R be the set of jobs that come after k and before ℓ in S. Let $\bar{S}$ be the schedule derived from S by starting job ℓ at time 0 after job j has completed, and by delaying the jobs in $\{k\} \cup R$ by p_ℓ time units. Let $p_R = \Sigma_{i \in R} p_i$. Job ℓ is late in both schedules, so it contributes $p_\ell(p_k + p_R)$ more to the cost of S than it does to the cost of $\bar{S}$. The jobs in $\{k\} \cup R$ contribute more to the cost of $\bar{S}$ than they do to the cost of S. The amount of the difference is at most $(p_k + p_R)\, p_\ell$, and it is strictly less than $(p_k + p_R)\, p_\ell$ if $b_k > 0$. If $b_k = 0$, $|\mathcal{E}|$ is smaller for $\bar{S}$ then it is for S, so S does not lexicographically minimize $(C(S), |\mathcal{E}|)$.

Case 2: $c_\ell(S) < 0$. Let R be the set of jobs that come before j and after ℓ in S. Let $\bar{S}$ be the schedule derived from S by completing job ℓ at time 0, right before job k starts, and by doing the jobs in $\{j\} \cup R$ p_ℓ time units earlier. Job ℓ is not late in $\bar{S}$, and job j is late in S. Therefore the cost of $\bar{S}$ is less than the cost of S. □

By Lemma 15, there is an optimal schedule in which all jobs in $\mathcal{E}$ precede all jobs in $\mathcal{L}$. A schedule is said to be *strongly optimal* if it is an optimal schedule in which all jobs in $\mathcal{E}$ precede all jobs in $\mathcal{L}$, the jobs in $\mathcal{E}$ are sequenced in EDD/SPT sequence (jobs are

sequenced by their due dates and ties broken by putting jobs with smaller processing times first), and the jobs in $\mathcal{L}$ are sequenced in EBD/LPT sequence (jobs are sequenced by their begin dates and ties broken by putting jobs with larger processing times first). Lemma 15 and Corollary 5 imply that there is a strongly optimal sequence.

A job i is said to be *dominated* if there is another job j such that $b_j \leq b_i$, $d_j \geq d_i$, and at least one of these inequalities is strict. A job is *undominated* if it is not dominated.

LEMMA 16. Let S be a strongly optimal schedule. Let the last early job in S be k and the first late job in S be ℓ, and let processing on job k complete at time zero in S (see Figure 4). Then in S,

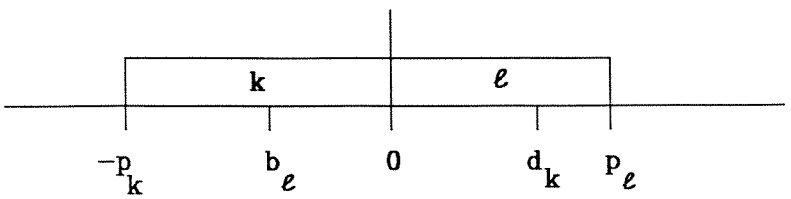
$$d_k < p_\ell \text{ and } b_\ell \geq -p_k, \tag{9}$$

$$\text{Either } d_k < d_\ell \text{ or } b_k \leq b_\ell, \tag{10}$$

$$\text{Either } k \text{ or } \ell \text{ is undominated,} \tag{11}$$

$$\begin{aligned} &\text{Let } j \in \{k, \ell\} \text{ and } j \text{ undominated.} \\ &\text{Then } b_i < b_j \Rightarrow i \in \mathcal{E} \text{ and } d_i > d_j \Rightarrow i \in \mathcal{L}. \end{aligned} \tag{12}$$

Figure 4.



Proof. **Proof of (9):** If $d_k \geq p_\ell$ then a cheaper schedule can be obtained by reversing the positions of k and ℓ. Let $b_\ell < -p_k$ and suppose that we reverse the positions of k and ℓ. Then ℓ is still late, so we save $p_\ell p_k$ on job ℓ. If $d_k > 0$ then the costs associated with k go up by less than $p_k p_\ell$ and the new schedule is less expensive than S. If $d_k = 0$ then the costs associated with k go up by exactly $p_k p_\ell$ and the two schedules have the same cost. But $|\mathcal{E}|$ has gone down by one, so S is not strongly optimal.

Proof of (10): Suppose that $d_\ell \leq d_k$ and $b_\ell < b_k$. Let $\bar{S}$ be the schedule obtained by interchanging k and ℓ. Since $k \in \mathcal{E}$ and $\ell \in \mathcal{L}$ we have $b_\ell < 0$ and $d_k \geq 0$. By (9),

$$\begin{aligned} C(S) - C(\bar{S}) &= p_\ell(-b_\ell) - p_k(p_\ell - d_k) \\ &> p_\ell(-b_\ell) - (d_k - b_\ell)(p_\ell - d_k) \\ &= d_k(d_k - d_\ell) \geq 0 . \end{aligned}$$

contradicting the optimality of S.

Proof of (11): Assume that both k and ℓ are dominated. Let $d_k < d_\ell$ and let j dominate ℓ. If $j \in \mathcal{E}$ then $d_k < d_j$, so j comes after k, a contradiction. If $j \in \mathcal{L}$ then j comes before ℓ because j dominates ℓ, a contradiction. Therefore $d_k \geq d_\ell$. By (10) we have $b_k \leq b_\ell$. Let i dominate k. This implies that i also dominates ℓ. If $i \in \mathcal{E}$ then i comes after k because i dominates k, a contradiction. If $i \in \mathcal{L}$ then i comes before ℓ because i dominates ℓ, a contradiction. Therefore either k or ℓ is undominated.

Proof of (12): Suppose that $b_i < b_j$ and $i \in \mathcal{L}$. Since $i \in \mathcal{L}$ we have $b_i \geq b_\ell$, so $j \neq \ell$, $j = k$, and $b_\ell < b_j$. By (10), $d_j < d_\ell$, which implies that ℓ dominates j, a contradiction. Suppose that $d_i > d_j$ and $i \in \mathcal{E}$. Since $i \in \mathcal{E}$ we have $d_i \leq d_k$, so $j \neq k$,

$j = \ell$, and $d_k > d_j$. By (10), $b_j \geq b_k$, which implies that k dominates j, a contradiction. □

We now give a pseudopolynomial-time algorithm for computing an optimal solution to Q. Recall that by Lemma 16, an optimal solution to the one-machine scheduling problem can be found by finding optimal sequences for each of the clusters and piecing them together. Therefore we assume that the poroblem has already been decomposed by cluster, and limit ourselves to instances of Q in which there is only one cluster.

The algorithm proceeds as follows. In Step 2 of the algorithm we select an undominated job j. For each such j, in Steps $3-6$ we find the best strongly optimal schedule in which j is the first late job, and in Steps $7-9$ we find the best strongly optimal schedule in which j is the last early job. In Step 10 we select the best of all of these schedules. By (11) of Lemma 16, this must be an optimal schedule.

We find the best strongly optimal schedule in which j is the first late job by focusing on the set $\mathcal{E}$. Note that because of strong optimality, j and $\mathcal{E}$ determine the entire schedule. Also note that by Lemma 2, minimizing the total earliness is equivalent to minimizing the total tardiness. Furthermore the set $\mathcal{E}$ determines the total earliness of a strongly optimal schedule. Thus in Steps 3–6 we search for the set $\mathcal{E}$ which has minimal total earliness, assuming that j is to be the first late job in the schedule.

If j is to be the first late job in a strongly optimal schedule then all jobs i such that $d_i > d_j$ are necessarily in $\mathcal{L}$, so they are excluded in Step 3. The jobs in $\mathcal{E}$ are sequenced in increasing order of their due dates in a strongly optimal schedule, so we index the remaining jobs in order of increasing due dates. A dynamic programming algorithm is then used. The states are ordered pairs (i,t). Corresponding to the state (i,t) is the following problem. Find a subset $\mathcal{E}(i,t)$ of $\{1, 2, \ldots, i\}$ with total processing time $t - B$ which, when the jobs in $\mathcal{E}(i,t)$ are sequenced in EDD fashion starting at time B, has minimal weighted earliness. The corresponding weighted earliness is denoted by $e(i,t)$.

Steps $7-10$ are similar. We assume that j is the last job in $\mathcal{E}$, and we search for a strongly optimal sequence in which this is the case. The set of jobs $\mathcal{L}(i,t)$ corresponds to a partial schedule in which the jobs are sequenced in EBD fashion starting at time t and ending at time F, and no jobs are early. The weighted tardiness of this partial schedule is $\tau(i,t)$.

The algorithm follows. The convex indicator function is $I(W) = 0$ if W is a true statement, and $I(W) = \infty$ if W is a false statement.

The Proportional–Weights Algorithm.

Step 1: Set $j \leftarrow 0$.

Step 2: Set $j \leftarrow j + 1$. If $j > J$ then go to Step 11. If j is dominated then repeat Step 2. Otherwise go to Step 3.

Step 3: Let $R = \{i \neq j : d_i \leq d_j\}$. If $\Sigma_{i \in R} p_i \leq b_j - B$ then set $e(j) = \infty$ and go to Step 7. Otherwise temporarily re-index the jobs so that $R = \{1, 2, \ldots, L\}$ and $d_i \leq d_{i+1}$ for all $i \in R$. Since there is only one cluster, $d_L > b_j$.

Step 4: Set $e(0,B) = 0$, and set $e(i,t) = \infty$ for all other (i,t). Set $\mathcal{E}(i,t) \leftarrow \phi$ for all (i,t).

Step 5: For each (i,t), $B + 1 \leq t \leq d_L$, $1 \leq i \leq L$, do the following in increasing order of t and i.

a. If $b_i < b_j$ then set $e(i,t) \leftarrow e(i-1,t-p_i) + I(t \leq d_i) + p_i(d_i - t)$.

b. If $b_i \geq b_j$ then set $e(i,t) \leftarrow min\,[e(i-1,t)\,,\, e(i-1,t-p_i) + I(t \leq d_i) + p_i(d_i - t)]$.

c. If $e(i,t) = e(i-1,t)$ then $\mathcal{E}(i,t) = \mathcal{E}(i-1,t)$; otherwise $\mathcal{E}(i,t) = \mathcal{E}(i-1,t-p_i) \cup \{i\}$.

Step 6: Let t^* be the value of t that minimizes $e(L,t)$ over all t, $b_j < t \leq d_L$. Set $e(j) \leftarrow e(L,t^*)$ and $\mathcal{E}(j) \leftarrow \mathcal{E}(L,t^*)$. Restore the original indexing of the jobs.

Step 7: Let $R = \{i \neq j : b_i \geq b_j\}$. If $\Sigma_{i \in R} p_i < F - d_j$ then set $\tau(j) = \infty$ and go to Step 2. Temporarily re-index the jobs so that $R = \{1, 2, \ldots, L\}$ and $b_i \leq b_{i+1}$ for all $i \in R$.

Step 8: Set $\tau(L+1,F) = 0$, and set $\tau(i,t) = \infty$ for all other (i,t). Set $\mathcal{L}(i,t) = \phi$ for all (i,t).

Step 9: For each (i,t), $b_1 < t \leq F$, $1 \leq i \leq L$, do the following in decreasing order of t and i.

a. If $d_i > d_j$ then set $\tau(i,t) \leftarrow \tau(i+1,t+p_i) + I(t > d_i) + p_i(t - d_i)$.

b. If $d_i \leq d_j$ then set $\tau(i,t) \leftarrow min\,[\tau(i+1,t)\,,\ \tau(i+1,t+p_i) + I(t > d_i) + p_i(t - d_i)]$.

c. If $\tau(i,t) = \tau(i+1,t)$ then $\mathcal{L}(i,t) = \mathcal{L}(i+1,t)$; otherwise $\mathcal{L}(i,t) = \mathcal{L}(i+1,t+p_i) \cup \{i\}$.

Step 10: Let t^* be the value of t that minimizes $\tau(1,t)$ over all t, $b_1 < t \leq d_j$. Set $\tau(j) \leftarrow \tau(1,t^*)$ and $\mathcal{L}(j) \leftarrow \mathcal{L}(1,t^*)$. Restore the original indexing of the jobs. Go to Step 2.

Step 11: The cost of the optimal schedule is

$$min_{j:j\ is\ undominated}\ min\left[\tau(j)\,,\ e(j) + \frac{1}{2}\left[\sum_i p_i\right]^2 + \frac{1}{2}\sum_i p_i^2 + \sum_i p_i(B - d_i)\right].$$

If $\tau(j^*)$ is the optimal cost then an optimal schedule is given by the jobs in $\mathcal{J}\backslash[\{j^*\} \cup \mathcal{L}(j^*)]$ in EDD sequence, followed by job j^*, followed by the jobs in $\mathcal{L}(j^*)$ in EBD sequence. If $e(j^*) + \frac{1}{2}\left[\sum_i p_i\right]^2 + \frac{1}{2}\sum_i p_i^2 + \sum_i p_i(B - d_i)$ is the optimal cost then an optimal schedule is given by the jobs in $\mathcal{E}(j^*)$ in EDD sequence, followed by job j^*, followed by the jobs in $\mathcal{J}\backslash[\{j^*\} \cup \mathcal{E}(j^*)]$ in EBD sequence.

The proof that the Proportional-Weights Algorithm computes an optimal schedule makes use of the following two lemmas. In the proofs of these two lemmas we assume that (1) holds. Recall that this assumption can be made without loss of generality, and that $\Sigma_i p_i = g(F) = F - B$.

LEMMA 17. In Step 6 let $e(j) < \infty$ and let $\mathcal{E} = \mathcal{E}(L,t^*)$, and let $\mathcal{L} = \mathcal{J}\backslash(\mathcal{E} \cup \{j\})$. Let S be the schedule created by first sequencing the jobs in $\mathcal{E}$ in EDD sequence, followed by job j, followed by the jobs in $\mathcal{L}$ in EBD sequence. Then in schedule S, no job in $\mathcal{L}$ is early.

PROOF. The condition in Step 3 implies that $d_j - B < p_j + \Sigma_{i\in R}\, p_i = \Sigma_{i:d_i\le d_j}\, p_i \le g(d_j)$. The definition of z implies that $z \le d_j$. The algorithm implies that the completion time of j in S is $c_j(S) > d_j \ge z$. Therefore if $i \in \mathcal{L}$ and $b_i \le d_j$ then i is late, and if $i \in \mathcal{L}$ and $b_i > d_j \ge z$ then all jobs m that follow i satisfy $b_m \ge b_i$, so Corollary 7 implies that i is not early. □

LEMMA 18. In Step 10 let $\tau(j) < \infty$, $\mathcal{L} = \mathcal{L}(1,t^*)$, and $\mathcal{E} = \mathcal{J}\backslash(\mathcal{L} \cup \{j\})$. Let S be the schedule created by first sequencing the jobs in $\mathcal{E}$ in EDD sequence, followed by job j, followed by the jobs in $\mathcal{L}$ in EBD sequence. Then in schedule S, no job in $\mathcal{E}$ is late.

PROOF. We claim that $b_j \le B$. The condition in Step 7 implies that $F - b_j \le p_j + \Sigma_{i\in R}\, p_i$ $= \Sigma_{i:b_i\ge b_j}\, p_i$, so $b_j - B \ge \Sigma_{i:b_i<b_j}\, p_i \ge g(b_j)$. Since there is only one cluster, either $b_j \le b_i$ $\forall$ i, or $g'(b_j + \epsilon) > 1$ for all sufficiently small $\epsilon > 0$. If $b_j \le b_i$ $\forall$ i then $g(b_j) = 0$ and $z \ge b_j$. If $b_j > b_i$ for some i then $b_j > B$, and the definition of z implies that $b_j \le z$. This proves the claim.

The algorithm implies that the time when j starts in schedule S is $s_j \le b_j \le z$. Therefore if $i \in \mathcal{E}$ and $d_i \ge b_j$ then i is not late in S, and if $i \in \mathcal{E}$ and $d_i < b_j \le z$ then all jobs m that precede i satisfy $d_m \le d_i$, so Corollary 7 implies that i is not late. □

THEOREM 19. The Proportional-Weights Algorithm computes an optimal schedule for P .

PROOF. In Step 6 of the algorithm, let $\mathcal{E} = \mathcal{E}(L,t^*)$. If $e(L,t^*) < \infty$, consider the schedule S obtained by first sequencing the jobs in $\mathcal{E}$ in EDD sequence, followed by job j, followed by the rest of the jobs in EBD sequence. The algorithm implies that the jobs in $\mathcal{E}$ are not late and that j is not early. Lemma 17 implies that the jobs that are not in $\mathcal{E} \cup \{j\}$ are not early. Therefore Lemma 2 implies that the weighted tardiness of these schedules is correctly calculated in Step 11. Similar statements apply to Steps 7 through 10, with Lemma 18 replacing Lemma 17. Therefore the finite costs considered in Step 11 correspond to actual costs of feasible schedules.

Assume that for some j the jobs are indexed according to Step 2, and that $\mathcal{E}(i,t)$ is a subset of $\{1, 2, \ldots, i\}$. We think of $\mathcal{E}(i,t)$ as partial schedules obtained by sequencing the elements of $\mathcal{E}(i,t)$ in an EDD sequence starting at time B . For each state (i,t) , problem $Q(i,t)$ is to select the set $\mathcal{E}(i,t)$ whose corresponding schedule has minimal earliness, subject to the constraint that none of the jobs in $\mathcal{E}(i,t)$ be late and that $\Sigma_{j \in \mathcal{E}(i,t)} p_j = t - B$. An induction argument based on Steps 4 and 5 implies that $e(i,t)$, as computed by the algorithm, is the optimal solution to problem $Q(i,t)$.

Suppose that S is a strongly optimal schedule. Let k be the last job in S that is not late, and let ℓ be the first strictly late job in S. Then ℓ comes immediately after k in S. By (11), either k or ℓ is undominated. If ℓ is undominated then the total earliness of S is the cost of a feasible solution to problem $Q(k, B + \Sigma_{i \leq k} p_i)$. Therefore the total earliness of S is greater than or equal to $e(k, \Sigma_{i \leq k} p_i)$. By Step 11 and Lemma 2, $C(S) \geq$ the cost of the schdule computed by the algorithm. A similar argument applies to the case in which ℓ is dominated, but k is not. □

The sets $\mathcal{E}(j,t)$ and $\mathcal{L}(j,t)$ were introduced into the algorithm for ease of exposition. The more traditional approach is more efficient. In this approach pointers record the choices made in Steps 5, 6, 9, 10, and 11. After the cost of an optimal schedule has been computed in Step 11, the sets $\mathcal{E}$ and $\mathcal{L}$ can be recovered using a traditional backtracking computation. When the traditional approach is used each iteration of steps 5 and 9 executes in constant time, so the algorithm clearly runs in $O(J^2 \Sigma_j p_j)$ time.

SECTION 7: AN NP–HARDNESS PROOF.

In this section we show that Q is NP-Hard.

THEOREM 20. Q is NP–Hard.

PROOF. We reduce the Subset Sum Problem (Garey and Johnson 1979) to Q. An instance of the Subset Sum Problem is determined by a set of integers $\{p_i : 1 \le i \le J\}$ and an integer M. The task is to find a subset $\{p_i : i \in Q\}$ of the integers such that $\Sigma_{i \in Q} p_i \le M$ and $\Sigma_{i \in Q} p_i$ is as large as possible.

Given an instance of the Subset Sum Problem we construct an instance of Q as follows. The job set is $J = \{0, 1, \ldots, J\}$. The processing time of job i, $i > 0$, is p_i. The processing time of job 0 is $p_0 = (\Sigma_{i \ge 1} p_i)^2$. The due dates are $d_0 = p_0 + M$ and $d_i = p_0 + \frac{1}{2} p_i$, $1 \le i \le J$. The start time is $B = 0$. The construction is clearly polynomial.

In a schedule for Q, let $\mathcal{E}$ be the set of jobs that come before 0 and let $\mathcal{L}$ be the set of jobs that come after 0. Clearly the jobs in $\mathcal{E}$ are all early and the jobs in $\mathcal{L}$ are all late. Let $p_{\mathcal{E}} = \Sigma_{i \in \mathcal{E}} p_i$ and $p_{\mathcal{L}} = \Sigma_{i \in \mathcal{L}} p_i$. Assuming that $\mathcal{L} = \{1, 2, \ldots, L\}$, the cost of the schedule is

$$p_0\left[p_{\mathcal{E}} - M\right]^+ + \sum_{i\in\mathcal{L}} p_i \left[p_{\mathcal{E}} + \sum_{j<i} p_j + \tfrac{1}{2} p_i\right]$$

$$- p_0\left[p_{\mathcal{E}} - M\right]^+ + \tfrac{1}{2} p_{\mathcal{L}}^2 + p_{\mathcal{L}} p_{\mathcal{E}}$$

$$= p_0\left[p_{\mathcal{E}} - M\right]^+ + \tfrac{1}{2}(p_{\mathcal{E}} + p_{\mathcal{L}})^2 - \tfrac{1}{2} p_{\mathcal{E}}^2 .$$

Since $p_0 > \frac{1}{2}(p_{\mathcal{E}} + p_{\mathcal{L}})^2$, no optimal sequence will have $p_{\mathcal{E}} > M$. Therefore the problem of finding an optimal schedule for Q is equivalent to the problem of finding a set $\mathcal{E} \subset \mathcal{J}$ that maximizes $p_{\mathcal{E}}$ subject to $p_{\mathcal{E}} \leq M$. □

SECTION 8: CONCLUSIONS AND EXTENSIONS.

We have shown the weighted tardiness scheduling problem with proportional weights to be NP-Hard, and have provided a pseudo-polynomial time algorithm for it. The running time of the algorithm is quadratic in the number of jobs and linear in the length of the planning horizon, so it should be efficient enough for use in many real-world applications.

Previously the only class of single-machine weighted tardiness scheduling problems for which polynomial-time algorithms existed were the problems with processing times of one, or problems with agreeable weights, i.e., $w_i \geq w_j$ whenever $p_i < p_j$. The weights we use ($w_i = p_i$ for all i) are also monotonic with the processing times, but in the opposite direction. It would be interesting to know whether pseudo-polynomial time algorithms exist for systems in which the weights are, for example, an increasing affine function of the processing times. It would also be interesting to attempt to extend these results to systems with parallel machines and to systems with priority classes.

The earliest-gamma-date scheduling rules work very well when the weights are approximately proportional to the processing times. They also work very well when there is very little tardiness. However they perform poorly when there is a lot of tardiness and the weights are severely disproportionate to the processing times. Of the earliest-gamma-date scheduling rules, the best one has $\gamma_i = (b_i + d_i)/2$.

Several possible extensions of the worst-case effectiveness results for the earliest-gamma-date scheduling rules come to mind. We have provided theoretical bounds on the cost peformance of earliest-gamma-date scheduling rules only when the weights are precisely proportional to the processing times. If this is only approximately true then a finite bound of the type we have presented herein does not exist, as the following example shows.

EXAMPLE 4.

$$J = \{1, 2, \ldots, 2N\}$$

$$p_i = 1$$

$$w_i = \begin{cases} 1 - \epsilon & 1 \le i \le N \\ 1 + \epsilon & N < i \le 2N \end{cases}$$

$$d_i = \begin{cases} N & 1 \le i \le N \\ N + \epsilon & N < i \le 2N \end{cases}$$

We have:

$$C(E\gamma D) = \sum_{1 \le i \le N} (1 + \epsilon)((N + i) - (N + \epsilon))$$

$$= \frac{N(N+1)}{2} + \frac{N(N-1)}{2}\epsilon - N\epsilon^2$$

$$C(Opt) = \sum_{1 \le i \le N} (1 - \epsilon)((N + i) - N)$$

$$= \frac{N(N+1)}{2} - \frac{N(N+1)}{2}\epsilon$$

$$\frac{C(E\gamma D) - C(Opt)}{\Sigma_i p_i^2} = \frac{N^2\epsilon - N\epsilon^2}{2N},$$

which tends to infinity as N tends to infinity.

However we conjecture that a finite upper bound on $[C(E\gamma D) - C(Opt)]/[\Sigma_i p_i^2 + C(Opt)]$ could be obtained for general weights, if the bound is a function of ϵ.

Another possible extension would be to allow arbitrary job release dates. In this case, a natural heuristic would be to use $E\gamma D$ sequencing among currently available jobs. The following example shows that our results cannot be extended to this case.

EXAMPLE 5.

$$J = \{0, 1, \ldots, N^2\}$$

$$p_i = \begin{cases} N, & i = 0 \\ 1, & i > 0 \end{cases} = w_i$$

$$d_i = \begin{cases} \infty, & i = 0 \\ i + 1, & i > 0 \end{cases}$$

$$r_i = \begin{cases} 0, & i = 0 \\ 1, & i > 0 \end{cases}$$

$$Opt = \{Idle\ until\ time\ 1;\ 1, 2, \ldots, N^2, 0\}$$

$$C(Opt) = 0$$

$$Heuristic = \{0, 1, 2, \ldots, N^2\}$$

$$C(Heuristic) = N^2(N-1) = N^3 - N^2$$

$$\Sigma_i p_i^2 = N^2 + N^2 = 2N^2$$

$$\frac{C(Heuristic) - C(Opt)}{\Sigma_i p_i^2} = \frac{N}{2} - \frac{1}{2},$$

which tends to infinity as N tends to infinity.

Clearly, the ratio $[C(Heuristic) - C(Opt)]/[\Sigma_i p_i^2 + C(Opt)]$ cannot have a finite bound. We are currently investigating other possible extensions such as systems with priority classes.

ACKNOWLEDGEMENT.

The authors gratefully acknowledge the valuable input of the referees. This research was supported by the National Science Foundation under grant number DMC–8451984, and by AT&T Information Systems and DuPont Corporation.

REFERENCES

Baker, K.R. and L.E. Schrage (1978). "Finding an Optimal Sequence by Dynamic Programming: An Extension to Precedence-Related Tasks". *Operations Research* **26**, 111–120.

Conway, R.W., W.L. Maxwell, and L.W. Miller (1967). *Theory of Scheduling*, Addison-Wesley, p. 30.

Du, J. and J.Y. Leung (1988). "Minimizing Total Tardiness on One Machine is NP-Hard." Working paper, Computer Science Department, University of Texas at Dallas, Richardson Texas.

Eastman, W.L., S. Even, and I.M. Isaacs (1964). "Bounds for the Optimal Scheduling of n Jobs on m Processors." *Management Science* **11**, 268–279.

Emmons, H. (1969). "One Machine Sequencing to Minimize Certain Functions of Job Tardiness". *Operations Research* **17**, 701–715.

Fisher, M.L. (1976). "A Dual Algorithm for the One–Machine Scheduling Problem". *Mathematical Programming* **11**, 229–251.

Garey, M.R. and D.S. Johnson (1979). "Computers and Intractability: A Guide to the Theory of NP–Completeness." W.H. Freeman and Company, San Francisco, CA.

Kao, E.P.C. and M. Queyranne (1982). "On Dynamic Programming Methods for Assembly Line Balancing". *Operations Research* **30**, 375–390.

Kawaguchi, T. and S. Kyan (1986). "Worst Case Bound of an LRF Schedule for the Mean Weighted Flow-Time Problem." *SIAM Journal of Computing* **15**, 1119–1129.

Lawler, E.L. (1977). "A 'Pseudopolynomial' Algorithm for Sequencing Jobs to Minimize Total Tardiness". *Annals of Discrete Mathematics*, 1977, 331–342.

Lawler, E.L. (1979). "Efficient Implementation of Dynamic Programming Algorithms for Sequencing Problems". Report BW 106, Mathematisch Centrum, Amsterdam.

Lawler, E.L., J.K. Lenstra, and A.H.G. Rinnooy Kan (1982). "Recent Developments in Deterministic Sequencing and Scheduling: A Survey". in *Deterministic and Stochastic Scheduling*, editors Dempster et al, Reidel Publishing Company, 1982, pp. 35–73.

Lenstra, J.K., A.H.G. Rinnooy Kan, and P. Brucker (1977). "Complexity of Machine Scheduling Problems". in *Annals of Discrete Mathematics* vol. 1, editors Dempster et al, North Holland Publishing Company, 1977, pp. 343–362.

Lenstra, J.K. and A.H.G. Rinnooy Kan (1985). "Sequencing and Scheduling". in *Combinatorial Optimization: Annotated Bibliographies*, editors O'hEighertaigh et al, John Wiley & Sons, 1985, pp. 164–189.

Morton, T.E., R.M. Rachamadugu, and A. Vepsalainen (1984). "Accurate Myopic Heuristics for Tardiness Scheduling". Working Paper #36–83–84, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Schrage, L.E. and K.R. Baker (1978). "Dynamic Programming Solution of Sequencing Problems with Precedence Constraints". *Operations Research* **26**, 444–449.

Weiss, G. (1988a). "Approximation Results in Parallel Machines Stochastic Scheduling." Technical Report ISyE J–88–3, Georgia Institute of Technology, Atlanta, Georgia.

Weiss, G. (1988a). "Turnpike Optimality of Smith's Rule in Parallel Machines Stochastic Scheduling." Technical Report ISyE J–88–9, Georgia Institute of Technology, Atlanta, Georgia.