NEW LOWER AND UPPER BOUNDS FOR SCHEDULING AROUND A SMALL COMMON DUE DATE

J. A. HOOGEVEEN
Eindhoven University of Technology, Eindhoven, The Netherlands

H. OOSTERHOUT
Tilburg University, Tilburg, The Netherlands

S. L. VAN DE VELDE
University of Twente, Enschede, The Netherlands

(Received October 1990; revision received February 1992; accepted September 1992)

We consider the single-machine problem of scheduling n jobs to minimize the sum of the deviations of the job completion times from a given small common due date. For this NP-hard problem, we develop a branch-and-bound algorithm based on Lagrangian lower and upper bounds that are found in $O(n \log n)$ time. We identify conditions under which the bounds concur; these conditions can be expected to be satisfied by many instances with n not too small. In our experiments with processing times drawn from a uniform distribution, the bounds concur for $n \geqslant 40$. For the case where the bounds do not concur, we present a refined lower bound that is obtained by solving a subset-sum problem of small dimension to optimality. We further develop a 4/3-approximation algorithm based upon the Lagrangian upper bound.

The just-in-time concept for manufacturing has induced a new type of machine scheduling problem in which both early and tardy completions of jobs are penalized. We consider the following single-machine scheduling problem that is associated with this concept.

A set of n independent jobs has to be scheduled on a single machine, which can handle no more than one job at a time. The machine is assumed to be continuously available from time zero onwards only. Job J_j requires processing during a given uninterrupted time p_j and should ideally be completed at a given due date d_j. Without loss of generality, we assume that the processing times and the due dates are integral. We assume furthermore that the jobs are indexed in order of nonincreasing processing times. A *schedule* σ defines for each job J_j a completion time C_j, such that the jobs do not overlap in their execution. The earliness and tardiness of J_j are defined as $E_j = \max\{d_j - C_j, 0\}$ and $T_j = \max\{C_j - d_j, 0\}$, respectively. The just-in-time philosophy is reflected in the objective function $f(\sigma) = \sum_{j=1}^{n} (\alpha_j E_j + \beta_j T_j)$, where the deviation of C_j from d_j is penalized by either α_j or β_j, depending on whether J_j is early or tardy for $j = 1, \ldots, n$. For a review of problems with this type of objective function, we refer to the survey by Baker and Scudder (1990).

An important subclass contains problems with a due date d that is common to all jobs. The common due date is either specified as part of the problem instance, or is a decision variable that has to be optimized simultaneously with the job sequence. As the first job may start later than time zero, the optimal schedule is identical for both problems unless the common due date d is restrictively small. The first variant is therefore referred to as the restricted problem and the second variant as the unrestricted problem.

Bagchi, Chang and Sullivan (1987) propose a branch-and-bound approach for the restricted variant with all earliness penalties equal to α and all tardiness penalties equal to β. Szwarc (1989) presents a branch-and-bound approach for the case that $\alpha = \beta$. These branch-and-bound algorithms are able to solve instances up to 25 jobs. Sundararaghavan and Ahmed (1984) present an approximation algorithm for the case $\alpha = \beta$ that shows a remarkably good performance from an empirical point of view. Lee and Liman (1992) present an approximation algorithm for the case $\alpha = \beta$ with *performance guarantee* 3/2; this means that for any instance their approximation algorithm produces a solution with a value no more than 3/2 times the optimal solution value. Hall, Kubiak and Sethi (1991) and Hoogeveen and van de Velde (1991)

Subject classifications: Production/scheduling: one-machine scheduling problem. Programming: Lagrangian relaxation.
Area of review: MANUFACTURING, OPERATIONS AND SCHEDULING.

0030-364X/94/4201-0102 $01.25

establish the NP-hardness of the problem, even if $\alpha = \beta$, thereby justifying the enumerative and approximative approaches. Hall, Kubiak and Sethi propose furthermore a pseudopolynomial time algorithm running in $O(n \sum_{j=1}^{n} p_j)$ time and space for general α and β, and provide computational results for instances up to 1,000 jobs for the case $\alpha = \beta$. Their experiments show that the algorithm is mainly limited by space, not time.

We present a branch-and-bound algorithm for the case $\alpha = \beta$. Using Lagrangian relaxation, we find new lower and upper bounds in $O(n \log n)$ time. All our computational experiments with processing times drawn from different distributions over the integers $1, \ldots, 100$ exhibit that for $n \geq 50$ the bounds always concur.

For the case that these bounds do not concur, we present a refinement of the lower bound, which is obtained by solving a subset-sum problem to optimality by a pseudopolynomial algorithm. This can be done very fast, because the subset-sum problem in our application is of a considerably smaller dimension than the common due date problem. Computational experiments show that, if any, only a small number of nodes are examined in the branch-and-bound algorithm.

In addition, we develop a heuristic that is based upon the Lagrangian upper bound with performance guarantee 4/3. This means that for any instance the heuristic produces a solution with a value no more than 4/3 times the optimal value.

This paper is organized as follows. In Section 1, we review Emmons's matching algorithm (Emmons 1987) for the unrestricted variant of the common due date problem with general α and β; it is needed in Section 2, where we develop a lower bound based upon Lagrangian relaxation for the restricted variant with $\alpha = \beta$. In Section 3, we use the insight gained in Section 2 to develop a heuristic for the restricted variant. In Section 4, we show that this heuristic has a performance guarantee 4/3. In Section 5, we describe the branch-and-bound algorithm, and in Section 6, we present some computational results. Finally, we briefly indicate to what extent the analysis applies to the case $\alpha \neq \beta$.

1. EMMONS'S MATCHING ALGORITHM FOR THE UNRESTRICTED PROBLEM

Kanet (1981) presents an $O(n \log n)$ algorithm for the unrestricted variant with $\alpha = \beta$. Bagchi, Chang and Sullivan (1987) and Emmons (1987) propose $O(n \log n)$ algorithms for the case that $\alpha \neq \beta$. We briefly review the concepts of Emmons's matching algorithm, because they provide the insight needed for the subsequent sections.

Theorem 1. (Kanet) *No optimal schedule has idle time between the execution of the jobs.*

Theorem 2. (Kanet) *There is an optimal schedule for the unrestricted variant in which the due date d coincides with the start time or completion time of the job with the smallest processing time.*

Emmons's matching algorithm is based upon the concept of positional weights. The scheduling problem then reduces to an assignment problem where jobs have to be assigned to positions. The cost of assigning J_j to the kth early position is equal to $\alpha(k - 1)p_j$; the cost of assigning J_j to the kth tardy position is equal to $\beta k p_j$. The assignment problem is solved in $O(n \log n)$ time by matching the job that has the jth largest processing time with the position that has the jth smallest weight for $j = 1, \ldots, n$.

Emmons's matching algorithm shows that in any optimal schedule the jobs completed before or at d are scheduled in order of nonincreasing processing times and the jobs started at or after d are in order of nondecreasing processing times. Due to this structure, optimal schedules are said to be *V-shaped.*

Optimal schedules for the restricted variant have the same structure, albeit there may be one job that is scheduled around d. For this particular job, it holds that the early jobs have equal or larger processing times, or the tardy jobs have equal or larger processing times.

2. A NEW LOWER BOUND FOR THE RESTRICTED VARIANT

We look upon this NP-hard problem as an "easy" problem complicated by the "nasty" constraint that the machine is only available from time zero onwards. If this constraint were not present, then the problem could easily be solved through Emmons's algorithm. This is exactly the approach Szwarc follows in determining a lower bound. The structure of the problem, however, suggests that the technique of Lagrangian relaxation might be more successful. We remove the nasty constraint and put it into the objective function, weighted by a nonnegative Lagrangian multiplier. The resulting problem is easy to solve. It will be referred to as the Lagrangian problem; its solution provides a lower bound for the original problem.

The nasty constraint can be formulated as $W \leq d$, where W denotes the total amount of work that is processed up to time d. If we introduce a Lagrangian multiplier $\lambda \geq 0$ and bring this constraint weighted by λ into the objective function, then we get the following Lagrangian problem, referred to as problem $\mathbf{L}_\lambda$: Find the value $\mathbf{L}(\lambda)$, which is the minimum of

$$\sum_{j=1}^{n} (E_j + T_j) + \lambda(W - d),$$

for a given $\lambda \geq 0$. Obviously, $\mathbf{L}(\lambda)$ is a lower bound for the original problem. There are two questions that immediately arise: Given a value of λ, can $\mathbf{L}(\lambda)$ be determined in polynomial time? If so, can the value λ^* that maximizes the lower bound $\mathbf{L}(\lambda)$ be found in polynomial time? The latter problem is referred to as the *Lagrangian dual problem*. The following two theorems provide affirmative answers to both questions.

Theorem 3. *For any given λ, the Lagrangian problem is solved by applying Emmons's matching algorithm with the weights of the early positions increased by λ.*

Proof. Straightforward arguments show that there exists an optimal schedule for the Lagrangian problem in which some job is completed exactly on time d. Hence, there is an optimal schedule with $W = \sum_{J_j \in \varepsilon} p_j$, where ε denotes the set of jobs that are scheduled in the early and just-in-time positions. The Lagrangian objective function can then alternatively be written as

$$\left\{ \sum_{j=1}^{n} (E_j + T_j) + \sum_{J_j \in \varepsilon} \lambda p_j \right\} - \lambda d.$$

Since the last term is a constant for a given λ, we need to minimize only the expression inside the braces. This is achieved by applying Emmons's matching algorithm to the case where the weight of the kth early position is equal to $k - 1 + \lambda$.

Theorem 4. *The optimal value λ^*, that is, the value that maximizes the Lagrangian lower bound, is equal to the index λ for which*

$$\sum_{j=0}^{\lfloor (n-\lambda)/2 \rfloor} p_{\lambda+2j} > d \geq \sum_{j=0}^{\lfloor (n-\lambda-1)/2 \rfloor} p_{\lambda+1+2j},$$

where $\lfloor x \rfloor$ denotes the largest integer smaller than or equal to x. If no such index exists, then $\lambda^* = 0$.

Proof. Consider an arbitrary value λ. If λ is not integral, then all optimal schedules for $(\mathbf{L}_\lambda)$ have equal W. If λ is integral, then there are multiple optimal schedules with different W; these are found by breaking ties differently in Emmons's algorithm. Define for each integer λ ($\lambda = 0, \ldots, n$) the schedule $\sigma_\lambda^{\max}$ as the optimal schedule for the Lagrangian problem $(\mathbf{L}_\lambda)$ with W maximal for $\lambda = 0, \ldots, n$. We define $W_\lambda^{\min}$ and $W_\lambda^{\max}$ as the amount of work processed before time d in $\sigma_\lambda^{\min}$ and $\sigma_\lambda^{\max}$, respectively. Straightforward calculations show that $\sigma_\lambda^{\min}$ remains optimal if the Lagrangian multiplier is increased by ϵ with $0 \leq \epsilon \leq 1$; hence, we have that $\sigma_\lambda^{\min}$ is identical to $\sigma_{\lambda+1}^{\max}$ and $W_\lambda^{\min} = W_{\lambda+1}^{\max}$. This implies that $\mathbf{L}(\lambda)$ is a piecewise-linear and concave function of λ. The breakpoints correspond to the integral values $\lambda = 1, \ldots, n$, and the gradient of the function between the integral breakpoints λ and $\lambda + 1$ is equal to $W_\lambda^{\min} - d$ for $\lambda = 0, \ldots, n - 1$. The Lagrangian dual problem is therefore solved by putting λ^* equal to the index λ for which $W_\lambda^{\max} > d \geq W_\lambda^{\min}$. Due to the indexing of the jobs, the theorem follows.

Let σ^* be an optimal schedule for the Lagrangian dual problem. If $\lambda^* = 0$, then $\sigma^* = \sigma_0^{\min}$ is feasible for the original problem, and hence is optimal. Note that this also implies that $d \geq p_1 + p_3 + \ldots + p_n$ if n is odd, and $d \geq p_1 + p_3 + \ldots + p_{n-1}$ if n is even. This agrees with the result by Bagchi, Chang and Sullivan that the schedules $(J_1, J_3, \ldots, J_n, J_{n-1}, \ldots, J_2)$ and $(J_1, J_3, \ldots, J_{n-1}, J_n, \ldots, J_2)$ are optimal under the respective conditions.

In the remainder, we assume that $\lambda^* \geq 1$. Depending on whether $n - \lambda^*$ is odd or even, σ^* has the following structure. First, suppose that $n - \lambda^*$ is odd. Then the jobs $J_1, \ldots, J_{\lambda^*-1}$ occupy the last $\lambda^* - 1$ positions in σ^*, the pair $\{J_{\lambda^*}, J_{\lambda^*+1}\}$ occupies the first early position and the λ^*th tardy position, the pair $\{J_{\lambda^*+2}, J_{\lambda^*+3}\}$ occupies the second early position and the $(\lambda^* + 1)$th tardy position, and so on. Finally, the pair $\{J_{n-1}, J_n\}$ occupies the positions around the due date. Second, if $n - \lambda^*$ is even, then σ^* has the same structure, except that J_n is positioned between J_{n-2} and J_{n-1}, and is started somewhere in the interval $[d - p_n, d]$.

Proposition 1. *If there exists a schedule σ^* that is optimal for the Lagrangian dual problem in which the first job is started at time zero, then the Lagrangian lower bound $L(\lambda^*)$ is tight and σ^* is an optimal schedule for the original problem.*

Proof. In this case we have $L(\lambda^*) = \sum(E_j + T_j) + \lambda^*(W - d) = \sum(E_j + T_j) = f(\sigma^*)$.

If no such schedule σ^* exists, then there is a gap between the optimal value for the original problem and the Lagrangian lower bound. We get a better lower bound, however, by solving the *modified* Lagrangian problem, which is to find a schedule that minimizes

$$\sum_{j=1}^{n} |C_j - d| + \lambda^*(W - d) + |W - d|.$$

Clearly, the modified Lagrangian problem yields a lower bound for the original problem for any $\lambda^* \geq 1$.

Theorem 5. *The modified Lagrangian problem is solved by a schedule from among the optimal schedules for the Lagrangian dual problem that has minimal* $|W - d|$.

Proof. Suppose that π is a schedule that has minimal Lagrangian cost from among the optimal schedules for the modified Lagrangian problem; suppose further that π is not optimal for the Lagrangian dual problem. Then either the jobs are not assigned to the optimal set of positions, or there are at least two jobs J_i and J_j with $p_i > p_j$ that are not optimally assigned. As to the first case, assigning J_i to a position with smaller weight decreases the Lagrangian cost by at least p_i, while $|W - d|$ is increased by at most p_i. As to the second case, the interchange of J_i and J_j decreases the Lagrangian cost by at least $p_i - p_j$, while $|W - d|$ is increased by at most $p_i - p_j$. In either case, π is easily transformed into a schedule $\tilde{\pi}$ that is also optimal for the modified Lagrangian problem but that has a smaller Lagrangian cost than π. This contradicts the assumption that π has a minimal Lagrangian cost. Hence, π must be also optimal for the Lagrangian dual problem.

The problem of minimizing $|W - d|$ is transformed into a considerably smaller instance of subset-sum in the following way. Renumber the jobs such that $J_{k-1+\lambda^*}$ becomes J_k for $k = 1, \ldots, n - \lambda^* + 1$; n becomes equal to $n - \lambda^* + 1$; the jobs previously denoted by $J_1, \ldots, J_{\lambda^*-1}$ are now simply referred to as the "remaining" jobs. Hence, the jobs $\{J_{2k-1}, J_{2k}\}$ form a pair in the Lagrangian dual for $k = 1, \ldots, l$ with $l = 1, \ldots, \lfloor n/2 \rfloor$. Define a_j as the difference in processing time between the jobs of the jth pair ($j = 1, \ldots, l$), and define $D = d - W_{\lambda^*}^{\min}$. Remove the values a_j that are zero; suppose that m of them remain. Define $\mathscr{A}$ as the multiset containing the m remaining a_j values; let $a_{[j]}$ denote the jth largest element in $\mathscr{A}$.

If n is even, then the problem of minimizing $|W - d|$ is equivalent to determining a subset $A \subseteq \mathscr{A}$, whose sum is as close to D as possible. If n is odd, then an optimal schedule for the Lagrangian dual problem is optimal for the original problem in the case of $W \in [d - p_n, d]$. Finding such a schedule is equivalent to determining a subset $A \subseteq \mathscr{A}$ whose sum falls in the interval $[D - p_n, D]$. If no such subset exists, then the goal is to find a subset A whose sum is as close as possible to either $D - p_n$ or D. This problem, known as the optimization version of subset-sum, is NP-hard in the ordinary sense (Garey and Johnson 1979).

The instance of subset-sum can then be solved to optimality by dynamic programming requiring $O(mD)$ time and space. Note that $D \leq \sum a_j \leq p_{\max}$; hence, the subset-sum problem is of a smaller dimension than the underlying common due date problem.

3. A NEW UPPER BOUND FOR THE RESTRICTED VARIANT

Consider an optimal schedule for the Lagrangian dual problem. If $W \leq d$, then it is also a feasible schedule for the common due date problem; if $W > d$, then we defer the schedule to make it feasible. The analysis in the previous section suggests that we should look for an optimal schedule for the Lagrangian dual problem with $|W - d|$ minimal. Recall that $W = d$ is a sufficient condition for also having an optimal schedule for the common due date problem.

We develop an approximation algorithm for the common due date problem based upon Johnson's approximation algorithm (Johnson 1974) for subset-sum, which runs in $O(m)$ time after sorting.

Johnson's Algorithm

STEP 1. $\mathscr{A} = \varnothing$; $j \leftarrow 1$.

STEP 2. If $a_{[j]} \leq D$, then $\mathscr{A} \leftarrow \mathscr{A} \cup \{a_{[j]}\}$ and $D \leftarrow D - a_{[j]}$.

STEP 3. $j \leftarrow j + 1$; if $j \leq m$, then go to Step 2.

Using an approximation algorithm for subset-sum rather than an optimization algorithm does not affect the worst-case behavior (see Section 4). As to the empirical behavior, our computational results suggest that the loss in accuracy, if any, is small.

Furthermore, we can identify a class of instances for which **Johnson's Algorithm** always finds a solution value equal to the *target sum D*. This class comprises the instances possessing the so-called *divisibility property*; this class is important in our application, as many instances can be expected to belong to it.

Definition. A multiset of values $\{a_1, \ldots, a_m\}$ with $1 = a_1 \leq a_2 \leq \ldots \leq a_m$ is said to possess the divisibility property if for every j ($j = 1, \ldots, m$) and for every value $D \in \{1, 2, \ldots, \sum_{i=1}^{j} a_i\}$ there exists a subset $A \subseteq \{a_1, \ldots, a_j\}$ whose sum is equal to D.

Theorem 6. *A multiset of values* $\{a_1, \ldots, a_m\}$ *with* $1 = a_1 \leqslant a_2 \leqslant \ldots \leqslant a_m$ *possesses the divisibility property if and only if* $a_{j+1} \leqslant \sum_{i=1}^{j} a_i + 1$ *for* $j = 1, \ldots, n-1$.

Theorem 7. *If an instance of subset-sum satisfies the divisibility property, then* **Johnson's Algorithm** *finds a subset with a sum equal to D.*

In our application, each a_j is equal to the difference in processing times between two successive jobs in the shortest processing time order. If the number of jobs with different processing times is not too small, then the values a_j tend to be small. Hence, for a randomly generated instance the likelihood of possessing the divisibility property increases with the number of jobs.

Johnson's Algorithm always yields a subset with a sum no more than D. This handicap is overcome by also applying the algorithm to the target sum $\bar{D} = \sum_{j=1}^{m} a_j - D$ and taking the complement of the resulting subset with respect to $\mathscr{A}$. We use the subscripts 1 and 2 to distinguish the approximation from below and from above: A_1 and D_1 denote the resulting subset and the gap for the approximation from below, and A_2 and D_2 denote the resulting subset and the gap for the approximation from above.

If both $D_1 > 0$ and $D_2 > 0$, then we apply the following algorithm to derive feasible schedules for the common due date problem from the subsets A_1 and A_2.

Algorithm Transform

STEP 1. Consider A_1. Starting with $\sigma_{X^*}^{\min}$, interchange the jobs that correspond to $a_j \in A_1$ for $j = 1, \ldots, m$, thereby increasing W by a_j per interchange. Determine the schedule corresponding to A_2 in a similar fashion, starting from $\sigma_{X^*}^{\max}$. Let the resulting schedules be σ_1 and σ_2.

STEP 2. The schedule σ_1 is started at time D_1. Shift the schedule to the left until the first job is started at time 0, or until the number of jobs completed before or at d exceeds the number of jobs completed after d by two. Rearrange the jobs to make the schedule V-shaped again. The resulting schedule is denoted by $\tilde{\sigma}_1$.

STEP 3. The schedule σ_2 is started at time $-D_2$. Defer the schedule such that the first job is started at time zero, and rearrange the jobs to make the schedule V-shaped again; this schedule is denoted by $\tilde{\sigma}_2^0$. If some J_k is scheduled around d, then defer $\tilde{\sigma}_2^0$ until J_k is started exactly at d. Rearrange the jobs to make the schedule V-shaped; let the resulting schedule be $\tilde{\sigma}_2$.

We now present our approximation algorithm for the common due date problem; in the remainder, we refer to it as the **Even-Odd Heuristic**.

Even-Odd Heuristic

STEP 0. Given an instance of the common due date problem, solve the Lagrangian dual problem, and apply **Johnson's Algorithm** to the corresponding instance of subset-sum.

STEP 1. If $D_1 \leqslant D_2$, then apply **Algorithm Transform**; go to Step 5.

STEP 2. Let $Q = \{a_j \mid a_j \geqslant D_1\}$. If $Q \neq \{a_1\}$, then apply **Algorithm Transform**, and go to Step 5.

STEP 3. If $p_1 > d$, then apply **Algorithm Transform** to determine $\tilde{\sigma}_2^0$. Furthermore, solve the Lagrangian dual problem under the condition that J_1 and all the "remaining" jobs occupy the last positions; go to Step 5.

STEP 4. Solve the Lagrangian dual problem under the condition that J_1 and the "remaining" jobs are assigned to positions after d, and solve the Lagrangian dual problem with J_1 assigned to a position before d. Apply **Johnson's Algorithm** and **Algorithm Transform** to all these solutions.

STEP 5. Choose a schedule with minimal cost.

4. WORST-CASE BEHAVIOR

For any instance I of the common due date problem, let $EOH(I)$ denote the solution value determined by the **Even-Odd Heuristic**, and let $OPT(I)$ denote the optimal solution value. We define ρ as

$$\rho = \sup_I \frac{EOH(I)}{OPT(I)}.$$

In this section, we prove that $\rho \leqslant 4/3$, that is, the **Even-Odd Heuristic** has performance guarantee 4/3.

Suppose first that **Johnson's Algorithm** does not solve the corresponding instance of subset-sum to optimality, that is, D_1 or D_2 is not minimal. This means that we do not know the minimal value of $W - d$, and therefore cannot use the strengthened lower bound in our analysis.

Lemma 1. *If* **Johnson's Algorithm** *does not solve the resulting instance of subset-sum to optimality, then* $\rho \leqslant 8/7$.

Proof. A straightforward analysis shows that, if **Johnson's Algorithm** leaves a gap G that is not minimal, then at least three a_j values greater than G are involved; this means there are at least six jobs with processing times at least equal to $3G$, $2G$, $2G$, G, G, and 0, respectively. Furthermore, due to the structure of the solution of the Lagrangian problem, the $\lambda^* - 1$ remaining jobs must have processing times of at least $3G$.

First, assume that $D_1 \leqslant D_2$. Then we have for any instance I that

$$EOH(I) \leqslant f(\sigma_1) = L(\lambda^*) + \lambda^* D_1 \leqslant OPT(I) + \lambda^* D_1.$$

Inspecting $\sigma_{\lambda^*}^{\max}$, we see that $L(\lambda^*) \geqslant D_1(5 + 3\lambda^*(\lambda^* + 1)/2)$. Hence, $\rho \leqslant 1 + (2\lambda^*/(10 + 3\lambda^*(\lambda^* + 1))) \leqslant 8/7$ for any $\lambda^* \geqslant 1$.

Second, assume that $D_1 > D_2$. If D_1 is not minimal, then we use the above analysis and find $\rho \leqslant 8/7$. If D_1 is minimal, then D_2 is not. Consider an element $a_j \notin A_2$ and suppose that $a_j < D_1 + D_2$. This implies that

$$\bar{D} < \sum_{k \in A_2} a_k + a_j < \bar{D} + D_1;$$

as a consequence, the sizes of the elements in $\mathscr{A} - A_2 - \{a_j\}$ add up to a value between $D - D_1$ and D, contradicting the minimality of D_1. Hence, $a_j \geqslant D_1 + D_2$, and the above analysis can be applied to establish $\rho \leqslant 8/7$.

So, if **Johnson's Algorithm** does not give minimal values of D_1 and D_2, then we surely have $\rho \leqslant 4/3$. From now on, we assume that D_1 and D_2 are minimal; hence, we can now use the strengthened lower bound.

Lemma 2. *If $D_1 \leqslant D_2$, then $\rho \leqslant 4/3$.*

Proof. Again, we have that $EOH(I) \leqslant L(\lambda^*) + \lambda^* D_1$. Furthermore, from Theorem 5 it follows that $OPT(I) \geqslant L(\lambda^*) + D_1$. Every element $a_j \notin A_1$ must have a size $a_j \geqslant D_1 + D_2 \geqslant 2D_1$. Inspecting $\sigma_{\lambda^*}^{\max}$, we see that $L(\lambda^*) \geqslant \lambda^*(\lambda^* - 1)D_1$; this gives $\rho \leqslant 1 + ((\lambda^* - 1)/(1 + \lambda^*(\lambda^* - 1))) \leqslant 4/3$ for any $\lambda \geqslant 1$.

Now suppose that $D_1 > D_2$. It is easy to show $\rho = 4/3$ if there exists an element $a_k \geqslant D_1$ with $k \geqslant 3$. If no such element exists, then we consider the costs of all schedules determined by **Algorithm Transform**. To that end, we need an upper bound on $\Delta = f(\bar{\sigma}_2) - f(\sigma_2)$.

Proposition 2. *Suppose that the first job in σ_2 has a processing time no more than d. Then Δ is no more than the sum of the positional costs in $\bar{\sigma}_2$ of the last k jobs before d and the first $k + 1$ jobs after d, where k is the number of jobs that have been transferred from a position before d to a position after d.*

Proof. Without loss of generality, we assume that n is even; if not, then we add a dummy job with zero processing time. As a matter of convenience, renumber the jobs temporarily such that $J_1, \ldots, J_k$ are the jobs that are transferred from positions before d to positions after d (J_k is completed at time d), and $J_{2k}, \ldots, J_{k+1}, J_0$ are the first $k + 1$ jobs after d (J_{2k} is started at time d). Note that the jobs J_i and J_{k+1} ($i = 1, \ldots, k$) form a pair in the Lagrangian dual; hence, we must have that $\min\{p_i, p_{k+i}\} \geqslant \max\{p_{i+1}, p_{k+i+1}\}$ for $i = 1, \ldots, k - 1$.

Suppose that J_0 occupies position $\lambda^* + \mu$ in $\bar{\sigma}_2$ with $\mu \geqslant 0$. Twice the positional cost of the jobs $J_0, \ldots, J_{2k}$ in σ_2 is then equal to

$$\begin{aligned}
&2((\mu + 1)p_1 + \ldots + (\mu + k)p_k \\
&\quad + (\lambda^* + \mu)p_0 + \ldots + (\lambda^* + \mu + k)p_{2k}) \\
&\geqslant (\lambda^* + \mu)p_0 \\
&\quad + ((\lambda^* + \mu)p_0 + (\lambda^* + \mu + 1)p_{k+1} + 2p_1) \\
&\quad + \ldots + ((\lambda^* + \mu + k - 1)p_{2k-1} \\
&\quad + (\lambda^* + \mu + k)p_{2k} + 2kp_k) \\
&\geqslant (\lambda^* + \mu)p_0 + (\lambda^* + \mu + 1)(p_1 + p_{k+1}) \\
&\quad + \min\{p_1, p_{k+1}\} + (\lambda^* + \mu + 3)(p_2 + p_{k+2}) \\
&\quad + \min\{p_2, p_{k+2}\} + \ldots + (\lambda^* + \mu + 2k - 1) \\
&\quad \cdot (p_k + p_{2k}) + \min\{p_k, p_{2k}\}.
\end{aligned}$$

The last expression is exactly equal to the positional cost due to the jobs $J_0, \ldots, J_{2k}$ in $\bar{\sigma}_2$.

Lemma 3. *Suppose that a_1 and a_2 are the only elements larger than D_1. Then $\rho \leqslant 4/3$.*

Proof. First, suppose that $p_1 + p_3 \leqslant d$. Partition the jobs into two subsets: The first one is $\{J_3, \ldots, J_n\}$, the second one consists of J_1, J_2, and the remaining jobs. As $p_1 + p_3 \leqslant d$, it follows immediately from Proposition 3 that for σ_2 the sum of the positional costs of the jobs in $\{J_3, \ldots, J_n\}$ is at least equal to Δ. The sum of the positional costs of the jobs in the other subset is at least $(1 + \lambda^{*2})D_1 \geqslant 2\lambda^* D_1$. Hence, $OPT(I) \geqslant 2\lambda^* D_1 + \Delta$, implying that $\rho \leqslant 4/3$.

Second, suppose that $p_1 + p_3 > d$. As a_1 and a_2 are the only two elements greater than D_1, it follows immediately that $D_1 = d - p_1 - p_4 - p_5 - \ldots$ if $a_1 \geqslant a_2$, and that $D_1 = d - p_2 - p_3 - p_5 - \ldots$ otherwise; $D_2 = p_1 + p_3 + p_6 + \ldots - d$. An easy interchange argument, validated by the inequality $D_1 > D_2$, proves that $J_1, J_3, J_n, J_{n-1}, \ldots$, is an optimal schedule for the case that J_1 and J_3 are started before time d. Hence, we are done unless J_1 or J_3 is started at or after time

d in any optimal schedule. In this case, however, we impose the additional constraint to the common due date problem that J_1 or J_3 is started at or after time d. Consider the modified Lagrangian problem with such an additional constraint. Along the lines of the proof of Theorem 5, we can show that this problem is solved by an optimal schedule for the Lagrangian dual problem with J_1 or J_3 scheduled after d for which $|W - d|$ is minimal; this is exactly the schedule σ_1. We have therefore that $OPT(I) \geq L(\lambda^*) + D_1 \geq (\lambda^{*2} + 2)D_1$. As $EOH(I) \leq L(\lambda^*) + \lambda^* D_1$, we obtain $\rho \leq 1 + ((\lambda^* - 1)/(\lambda^{*2} + 2)) < 4/3$.

The analysis of the case that a_2 is the only element greater than D_1 proceeds along the same lines.

Lemma 4. *Suppose that $D_1 > D_2$, a_1 is the only element greater than D_1, and $p_1 > d$. Then $EOH(I) = OPT(I)$.*

Proof. An easy interchange argument, validated by the inequality $D_1 > D_2$, proves that in any optimal schedule J_1 is either started at time 0, or scheduled immediately before the remaining jobs. The inequality $D_1 > D_2$ also implies that Emmons's matching algorithm determines a feasible and hence optimal schedule for the case that J_1 and the remaining jobs are started at or after d.

If $p_1 \leq d$, then we solve both the Lagrangian dual problem with the additional constraint that J_1 and all remaining jobs are scheduled after d and the Lagrangian dual problem with the additional constraint that J_1 is scheduled before d.

Lemma 5. *Suppose that $D_1 > D_2$, that a_1 is the only element greater than D_1, and $p_1 \leq d$. Then we have $\rho \leq 4/3$.*

Proof. First, suppose that there is an optimal schedule in which J_1 and the remaining jobs are started at or after d. Suppose that solving the Lagrangian dual problem under the condition that J_1 and all remaining jobs are assigned to positions after d gives $\bar{\lambda}^*$, $\bar{D}_1$, and $\bar{D}_2$. If $\bar{\lambda}^* = 0$, then we have found an optimal schedule. If $\bar{\lambda}^* \geq 1$, then the schedule that corresponds to $\bar{D}_2$ must begin with J_2, J_3, and J_4; if not, then W does not sum up to $d + D_2$. Hence, we have $a_1 \geq p_4$. This gives $EOH(I) \leq L(\bar{\lambda}^*) + \bar{\lambda}^* \bar{D}_1$, $OPT(I) \geq L(\bar{\lambda}^*)$, and $L(\bar{\lambda}^*) \geq (3 + ((\bar{\lambda}^* + 1)(\bar{\lambda}^* + 2)/2)D_1$ from which $\rho \leq 4/3$ follows.

Second, suppose there is an optimal schedule in which J_1 or some remaining job is not started after d. The optimal solution for the Lagrangian dual problem with the additional constraint that J_1 or some remaining job is not started after d is such that J_1 is started before d and all the remaining jobs after d; this is easily proven by an interchange argument. Suppose that solving this Lagrangian problem gives $\bar{\lambda}^*$, $\bar{D}_1$, and $\bar{D}_2$. Consider the schedule σ that corresponds to $\bar{D}_2$. Since $\bar{\lambda}^* \geq \lambda^* + 1$, the first job after J_1 must be some J_k with $k \geq 4$; hence, we have $\bar{D}_1 \leq p_4$. The case $\bar{D}_1 \leq \bar{D}_2$ is easy to handle; assume therefore that $\bar{D}_1 > \bar{D}_2$. Along the lines of Lemma 3, it can then be proven that $\rho \leq 4/3$.

Theorem 8. *The* **Even-Odd Heuristic** *has a performance guarantee 4/3, and this bound can be approximated arbitrarily closely.*

Proof. The first part follows immediately from Lemmas 1–5. The following example, based upon the case that only $a_2 > D_1$, shows that we can get arbitrarily close to this bound. Let D be an arbitrary positive integer. There are $n = 2D + 6$ jobs $\{J_1, \ldots, J_n\}$ with processing times

$$p_1 = p_2 = p_3 = D^2 + 2D,$$

$$p_4 = p_5 = p_6 = D,$$

$$p_i = 1 \quad \text{for } i = 7, \ldots, 2D + 6,$$

and with common due date $d = 2D^2 + 5D$. The **Even-Odd Heuristic** gives the schedules $J_1, J_4, J_5, J_7, \ldots, J_n, J_6, J_3, J_2$ with J_1 started at time D^2, and $J_1, J_3, J_5, J_7, \ldots, J_n, J_6, J_4, J_2$ with J_1 started at time zero. Both schedules have cost $4D^2 + 18D$. The optimal schedule $J_1, J_3, J_7, \ldots, J_n, J_6, J_5, J_4, J_2$, has cost $3D^2 + 19D$, however. Hence, we get arbitrarily close to 4/3 by choosing D sufficiently large.

5. BRANCH AND BOUND

First, we solve the Lagrangian dual problem. If $\lambda^* = 0$, then $\sigma^* = \sigma_0^{\min}$ is an optimal solution for the common due date problem, and we are done. Otherwise, we determine upper bounds as described in Section 3; we also apply the heuristic presented by Sundararaghavan and Ahmed. If the lower and the best upper bound do not concur, then we solve the subset-sum problem to optimality by dynamic programming. If the bounds still do not concur, then we apply branch and bound.

For the design of the search tree we make use of the V-shapedness of optimal schedules. Assume that the jobs have been re-indexed in order of nonincreasing processing times. A node at level j $(j = 1, \ldots, n)$ of the search tree corresponds to a partial schedule in which the completion times of the jobs $J_1, \ldots, J_j$ are fixed. Each node at level j has at most $n - j$ descendants. In the kth descendant ($k = 1, \ldots, n - j$), J_k is started before d and the jobs $J_{j+1}, \ldots, J_{j+k-1}$ are to be completed after d. Given the partial schedule for $J_1, \ldots, J_j$, a partial schedule for $J_1, \ldots, J_{j+k}$ is easy to compute.

The algorithm that we propose is of the depth-first type. We employ an *active node* search: At each level we choose one node to branch from. We consistently choose the node, whose job has the smallest remaining index. A simple but powerful rule to restrict the growth of the search tree is the following. A node at level j ($j = 1, \ldots, n$) corresponding to some J_k can be discarded if another node at the same level corresponding to some J_l with $p_k = p_l$ has already been considered. This rule obviously avoids duplication of schedules.

In the nodes of the tree, we only compute the lower bound $L(\lambda^*)$; we neither solve the modified Lagrangian dual problem nor compute additional upper bounds.

6. COMPUTATIONAL RESULTS

We considered two types of instances, depending on the distributions used to generate the processing times. Computational experiments were performed with $d = \lfloor t \sum p_j \rfloor$ for $t = 0.1, 0.2, 0.3, 0.4$, respectively, and with the number of jobs ranging from 10 to 1,000. For each combination of n and t we generated 100 instances. The algorithm was coded in the computer language C; the experiments were conducted on a Compaq-386 personal computer.

Table I shows some of the results for instances with the processing times drawn from the discrete uniform distribution [1, 100]. The design of the table reflects our three-phase approach. The third column, #$O(n \log n)$, shows the number of times (out of 100) that the **Even-Odd Heuristic** finds a schedule with a cost equal to the Lagrangian lower bound $L(\lambda^*)$; this is the number of times that the common due date problem was provably solved to optimality in $O(n \log n)$ time. The fourth column, # DP, shows how many of the remaining instances were provably solved to

Table I
Computational Results

n	t	# $O(n \log n)$	# DP	Maximum # of Nodes	# Even-Odd Optimal	# SA Optimal	# LB Tight
10	0.1	66	20	12	72	77	86
10	0.2	69	20	22	72	58	89
10	0.3	68	23	22	68	59	93
10	0.4	82	1	40	85	62	85
20	0.1	81	12	94	84	51	94
20	0.2	94	5	167	94	43	99
20	0.3	99	0	320	100	42	99
20	0.4	99	1	0	99	35	100
30	0.1	100	0	0	100	50	100
30	0.2	98	2	0	98	51	100
30	0.3	100	0	0	100	57	100
30	0.4	100	0	0	100	68	100
40	0.1	100	0	0	100	63	100
40	0.2	100	0	0	100	64	100
40	0.3	100	0	0	100	63	100
40	0.4	100	0	0	100	54	100
50	0.1	100	0	0	100	72	100
50	0.2	100	0	0	100	63	100
50	0.3	100	0	0	100	69	100
50	0.4	100	0	0	100	75	100
100	0.1	100	0	0	100	81	100
100	0.2	100	0	0	100	86	100
100	0.3	100	0	0	100	78	100
100	0.4	100	0	0	100	78	100

optimality by dynamic programming applied to subsetsum. The fifth column, maximum # nodes, shows the maximum number of nodes needed by the branch-and-bound algorithm. The sixth column, # even-odd optimal, shows the number of times that the **Even-Odd Heuristic** found an optimal schedule. The seventh column, # SA optimal, gives the same information for the approximation algorithm presented by Sundararaghavan and Ahmed. The last column, # LB tight, shows the number of times that the lower bound (strengthened or not) was equal to the optimal solution value. All these instances, including those with 1,000 jobs, were solved in less than 1 second, if the jobs were already sorted in order of nondecreasing processing times.

Instances with a large number of different processing times can be expected to possess the divisibility property. In this sense, the success of the algorithm for instances generated from a uniform distribution may be due mainly to the divisibility property. We therefore applied our algorithm to instances for which the divisibility property was expected to play a less prominent role. We generated these instances in the following way. We partitioned the jobs into subsets, whereafter all jobs in the same subset got the same processing time, drawn from the uniform distribution [1, 100]. The results exhibited essentially the same pattern as for the first type of instances, albeit the number of jobs to reach a 100% score went up a little.

7. EXTENSIONS

The lower bound approach can be extended to the restricted variant of each problem that is solvable by Emmons's matching algorithm. The most important problem in this context is $1 \mid d_j = d \mid \sum (\alpha E_j + \beta T_j)$. Without loss of generality, we assume that α and β are integral and relatively prime. A similar analysis shows that the optimal value λ^* is the value $\lambda^* \in \{1, \ldots, n\beta\}$ for which $W_{\lambda^*}^{\max} \geq d > W_{\lambda^*}^{\min}$. Furthermore, Theorem 5 still holds. It is straightforward to develop a heuristic for the common due date problem with $\alpha \neq \beta$ by applying **Johnson's Algorithm** and **Algorithm Transform**; its worst-case performance, however, is still an open question.

ACKNOWLEDGMENT

anonymous referees for their helpful comments. The work of the first author was supported by a grant from The Netherlands Organization for Scientific Research (NWO).

REFERENCES

BAGCHI, U., Y. L. CHANG AND R. S. SULLIVAN. 1987. Minimizing Absolute and Squared Deviations of Completion Times With Different Earliness and Tardiness Penalties and a Common Due Date. *Naval Res. Logist.* **34,** 739–751.

BAKER, K., AND G. SCUDDER. 1990. Sequencing With Earliness and Tardiness Penalties: A Review. *Opns. Res.* **38,** 22–57.

EMMONS, H. 1987. Scheduling to a Common Due Date on Parallel Uniform Processors. *Naval Res. Logist.* **34,** 803–810.

GAREY, M. R., AND D. S. JOHNSON. 1979. *Computers and Intractability.* W. H. Freeman, San Francisco.

HALL, N. G., W. KUBIAK AND S. P. SETHI. 1991. Earliness-Tardiness Scheduling Problems, II: Deviation of Completion Times About a Restrictive Common Due Date. *Opns. Res.* **39,** 847–856.

HOOGEVEEN, J. A., AND S. L. VAN DE VELDE. 1991. Scheduling Around a Small Common Due Date. *Eur. J. Opnl. Res.* **55,** 237–242.

JOHNSON, D. S. 1973. Approximation Algorithms for Combinatorial Problems. *J. Comp. and Syst. Sci.* **9,** 256–278.

KANET, J. J. 1981. Minimizing the Average Deviation of Job Completion Times About a Common Due Date. *Naval Res. Logist. Quart.* **28,** 643–651.

LEE, C.-Y., AND S. D. LIMAN. 1992. Error Bound for the Heuristic on the Common Due Date Scheduling Problem. *ORSA J. Comput.* (to appear).

SUNDARARAGHAVAN, P. S., AND M. U. AHMED. 1984. Minimizing the Sum of Absolute Lateness in Single-Machine and Multimachine Scheduling. *Naval Res. Logist. Quart.* **31,** 325–333.

SZWARC, W. 1989. Single Machine Scheduling to Minimize Absolute Deviation of Completion Times From a Common Due Date. *Naval Res. Logist.* **36,** 663–673.