Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.A. Hoogeveen, H. Oosterhout, S.L. van de Velde

New lower and upper bounds for scheduling around a small common due date

Department of Operations Research, Statistics, and System Theory Report BS-R9030 October

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

New Lower and Upper Bounds for Scheduling Around a Small Common Due Date

J.A. Hoogeveen
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

H. Oosterhout
Department of Economics, Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands

S.L. van de Velde
School of Management Studies, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands

Suppose a set of n jobs has to be scheduled on a single machine, which can handle no more than one job at a time. The problem is to find a schedule that minimizes the sum of the deviations of the job completion times from a given common due date that is smaller than the sum of the processing times. This problem is known to be $\mathcal{NP}$-hard. There exists a pseudo-polynomial algorithm that is able to solve instances up to 1000 jobs. Branch-and-bound algorithms can solve instances up to only 25 jobs. We apply Lagrangian relaxation to find new lower and upper bounds that coincide for virtually all instances with n not too small. Both bounds are computed in $O(n \log n)$ time. For the case that these bounds do not concur, we present refinements of the bounds, which are obtained by solving a subset-sum problem to optimality by a pseudo-polynomial algorithm; this subset-sum problem is of considerably smaller dimension than the common due date problem. We also show how the lower bounding approach and the heuristic can be extended to the problem where all early completions are weighted by a common weight α and all late completions by a common weight β.

1980 Mathematics Subject Classification (1985 Revision): 90B35.
Key Words & Phrases: single-machine scheduling, common due date, Lagrangian relaxation, subset-sum, approximation algorithm, branch-and-bound.
Note: This paper has been submitted for publication.

1. Introduction

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

Report BS-R9030
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

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 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 ($d < \Sigma p_j$). Therefore, the first variant is referred to as the restricted problem and the second variant as the unrestricted problem.

We consider the restricted variant of the problem in which all earliness penalties are equal to α and all tardiness penalties are equal to β. Bagchi, Chang, and Sullivan (1987) propose a branch-and-bound approach for this problem, and 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. Hall, Kubiak and Sethi (1990) and Hoogeveen and Van de Velde (1990) establish the $\mathcal{NP}$-hardness of the problem, even if $\alpha = \beta$, thereby justifying the enumerative and approximative approaches. Furthermore, Hall et al. (1990) propose a pseudo-polynomial time algorithm running in $O(n\Sigma p_j)$ time and space, and provide computational results for instances up to 1000 jobs.

Their experiments, however, show that the space requirement rather than the time requirement limits the applicability of the algorithm. In general, there is always need for a branch-and-bound algorithm that solves instances for which the space requirement is prohibitive. We present a branch-and-bound algorithm that solves virtually *all* instances without branching. It is based upon new lower and upper bounds, which are computed in $O(n \log n)$ time. If these bounds do not concur, they can be refined by solving a subset-sum problem to optimality by a pseudo-polynomial algorithm. This can be done very fast, since the subset-sum problem in our application is of a considerably smaller dimension than the common due date problem. Hence, the branch-and-bound algorithm is more than competitive with the pseudo-polynomial algorithm for the common due date problem.

This paper is organized as follows. In Section 2, we review Emmons' matching algorithm (Emmons, 1987) to solve the unrestricted variant of the common due date problem with arbitrary α and β. In Section 3, we develop a lower bound based upon Lagrangian relaxation for the restricted variant with $\alpha = \beta$ and show how it can be extended to the case $\alpha \neq \beta$. In Section 4, we use the insight gained in Section 3 to develop a heuristic for the restricted variant. In Section 5, we describe some details of the branch-and-bound algorithm. Finally, in Section 6, we

present some computational results.

2. Emmons' matching algorithm for the unrestricted problem

Kanet (1981) presents an $O(n \log n)$ algorithm for the unrestricted variant with $\alpha = \beta$. Bagchi et al. (1987) and Emmons (1987) propose $O(n \log n)$ algorithms for the case $\alpha \neq \beta$. We prefer to analyze Emmons' matching algorithm, since this provides the insight needed for the subsequent sections. We first present some properties of an optimal schedule for both variants of the problem.

THEOREM 1. *No optimal schedule has idle time between the execution of the jobs.* □

THEOREM 2. *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' matching algorithm is based upon the concept of positional weights, which stems from the following observation. If J_j is early, then E_j is equal to the sum of the processing times of all early and just-in-time jobs scheduled after J_j. This implies that ΣE_j can be rewritten as the sum of the weighted processing times of the early and just-in-time jobs, where the weight of p_j is equal to the number of jobs scheduled before J_j. Therefore, the positional weight of position k before d (which is also the kth position in the schedule) is equal to $\alpha(k-1)$. The weights for the positions after the common due date are determined in the same way. The weight of the last tardy position (which is the last position in the schedule) is then equal to β, the weight of the second last position is equal to 2β, and so on. This scheduling problem reduces then 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 $j = 1, \ldots, n$.

3. A new lower bound for the restricted variant

We first analyze the restricted variant in which earliness and tardiness are equally weighted, that is, $\alpha = \beta$. Note that for this case the objective function $f(\sigma)$ can be written as $\Sigma_{j=1}^{n} |C_j - d|$.

We look upon this $\mathcal{NP}$-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' algorithm. This is exactly the approach Szwarc (1989) follows to determine 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

$$W \leqslant d,$$

where W denotes the total amount of work that is processed up to time d. If we introduce a Lagrangian multiplier $\lambda \geqslant 0$ and bring this constraint weighted by λ into the objective function, then we get the following Lagrangian problem, referred to as problem (L_λ): find the value $L(\lambda)$, which is the minimum of

$$\sum_{j=1}^{n} |C_j - d| + \lambda(W - d), \tag{L_λ}$$

for a given $\lambda \geqslant 0$. Obviously, $L(\lambda)$ is a lower bound for the original problem. There are two questions that immediately arise: Given a value of λ, can $L(\lambda)$ be determined in polynomial time? If so, can the value λ^* that maximizes the lower bound $L(\lambda)$ 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 a given λ, the Lagrangian problem is solved by applying Emmons' matching algorithm with the weights of the positions before d 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 $W = \Sigma_{J_j \in \mathcal{E}} p_j$, where $\mathcal{E}$ 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

$$\{\sum_{j=1}^{n} |C_j - d| + \sum_{J_j \in \mathcal{E}} \lambda p_j\} - \lambda d.$$

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' 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 lower bound, is equal to the index λ for which*

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

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 Emmons' matching algorithm yields a unique optimal schedule. If λ is integral, then there is a large number of optimal schedules, which can be generated by breaking ties in different ways. Define for each integer λ ($\lambda = 0, \ldots, n$) $\sigma_\lambda^{\min}$ as the schedule with the property that the work processed before time d is minimal among all optimal schedules for the Lagrangian problem (L_λ). In the same fashion, the schedule $\sigma_\lambda^{\max}$ is defined as the optimal schedule for the Lagrangian problem (L_λ) with a maximal amount of work processed before time d, 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 $\sigma_\lambda^{\min}$ is identical to $\sigma_{\lambda+1}^{\max}$ and that $W_\lambda^{\min} = W_{\lambda+1}^{\max}$. This implies that $L(\lambda)$ 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 $W_\lambda^{\max} \geqslant d > W_\lambda^{\min}$. 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 optimal. Note that this also implies that $d \geqslant p_1 + p_3 + \cdots + p_n$ if n is odd, and $d \geqslant p_1 + p_3 \cdots + p_{n-1}$ if n is even. This agrees with the observation by Bagchi et al. (1987) 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.

PROPOSITION 1. *If $\lambda^* \geqslant 1$, then the first job in any optimal schedule for the original problem must be started at time zero.* □

In the remainder, we assume that $\lambda^* \geqslant 1$. Depending on whether $n - \lambda^*$ is odd or even, σ^* has the following structure. First, suppose $n - \lambda^*$ 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 (λ^*+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 2. *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.* □

If no such schedule σ^* exists, then there is a gap between the optimal value for the original problem and the Lagrangian lower bound. This lower bound, however, can be strenghtened 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 if $\lambda^* \geqslant 1$.

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

PROOF. Suppose that π is optimal for the modified Lagrangian problem, but not for the Lagrangian dual problem. We show that π can be transformed without additional cost into a schedule $\bar{\pi}$ that is optimal for the Lagrangian dual problem by conducting pairwise interchanges. Let π_t be the schedule after t interchanges; hence, $\pi_0 = \pi$, and $\pi_T = \bar{\pi}$, for some

$T \geqslant 1$. Note that it is possible to specify a series of pairwise interchanges that lowers the Lagrangian cost $\Sigma|C_j - d| + \lambda^*(W - d)$ at every interchange. Consider two successive schedules π_t and π_{t+1}, and suppose that J_i and J_j with $p_i > p_j$ have been interchanged. The interchange must have decreased the Lagrangian cost by at least $p_i - p_j$, and may have increased the term $|W - d|$ by at most $p_i - p_j$. This means that every interchange does not increase $\Sigma|C_j - d| + \lambda^*(W - d) + |W - d|$. Therefore, $\bar{\pi}$ must also be optimal. □

Suppose that the first job in σ^* is not started at time zero. According to Proposition 1, we shift σ^* to ensure that it starts at time zero, thereby either making it feasible or decreasing its cost. Shifting σ^* implies that some jobs or parts of jobs are transferred to the other side of the due date. First, suppose $n - \lambda^*$ is odd. If σ^* starts before time zero, then shifting σ^* increases $\Sigma|C_j - d|$ by $\lambda^* + 1$ per unit of the first job that is transferred, by $\lambda^* + 3$ per unit of the second job that is transferred, and so on. If σ^* starts after time zero, then shifting σ^* decreases $\Sigma|C_j - d|$ by $\lambda^* - 1$ per unit for the first job that is transferred, by $\lambda^* - 3$ per unit for the second job that is transferred, and so on. As $W - d$ is weighted by λ^*, the scheduling cost of σ^* after shifting is equal to $L(\lambda^*) + \Delta_1 + 3\Delta_2 + \cdots$, where Δ_j is the amount of the jth job that has been transferred. Second, suppose that $n - \lambda^*$ is even. A similar analysis shows that the scheduling cost of σ^* after shifting is equal to $L(\lambda^*) + 2\Delta_2 + 4\Delta_3 + \cdots$, where Δ_j is the amount of the jth job that has been transferred.

PROPOSITION 3. *The cost of the schedule determined in Theorem 5 after shifting such that the first job is started at time zero, is equal to the strengthened lower bound if both $n - \lambda^*$ is odd and if either J_{n-1} or J_n is executed at time d.* □

However, in order to determine that schedule, we have to solve the tie-breaking problem. We will deal with this problem in the next section.

The lower bound approach can be extended to the restricted variant of the problem with $\alpha \neq \beta$. Without loss of generality, we assume that α and β are integral and relatively prime. A similar analysis shows that the optimal value λ^* can be determined as the value $\lambda^* \in \{1, \ldots, n\beta\}$ for which $W_{\lambda^*}^{\max} \geqslant d > W_{\lambda^*}^{\min}$. Furthermore, Theorem 5 still holds, but the strengthening of the lower bound is less meaningful, since the cost of transferring one unit of processing time of the first job to the other side of d has been increased from 1 to either α or β, in case $n - \lambda^*$ is odd.

4. A NEW UPPER BOUND FOR THE RESTRICTED VARIANT

We start with the case $\alpha = \beta$. The analysis in the previous section suggests to find an optimal schedule for the Lagrangian dual problem with minimal $|W - d|$. This requires the development of a tie-breaking rule in Emmons' matching algorithm that minimizes $|W - d|$. Such a schedule induces an approximate solution to the common due date problem. This schedule is provably optimal if $W = d$ or if the conditions of Proposition 3 are satisfied.

We show that the problem of minimizing $|W - d|$ boils down to solving the optimization version of the *subset-sum* problem. This problem will be defined below; it will henceforth be

referred to as the subset-sum problem. Although this problem is $\mathcal{NP}$-hard in the ordinary sense (Garey and Johnson, 1979), the instances occurring in our application virtually always belong to an easy-to-solve subclass if n is not too small. In general, the subset-sum problem is solvable by a dynamic programming procedure that requires significantly less effort than the $O(n \Sigma p_j)$ time and space algorithm for the common due date problem.

The problem of minimizing $|W - d|$ can be transformed into an instance of the subset-sum problem in the following way. Define $a_j = p_{2j-2+\lambda^*} - p_{2j-1+\lambda^*}$, for $j = 1, \ldots, \lfloor (n - \lambda^* + 1)/2 \rfloor$, and define $D = d - W_{\lambda^*}^{\min}$. Note that all $a_j \geqslant 0$. Remove the values a_j that are equal to zero; let m be the number of remaining values a_j, and let $\mathcal{A}$ be the multiset that contains the values $a_1, \ldots, a_m$. First, suppose that $n - \lambda^*$ is odd. The problem of minimizing $|W - d|$ is then equivalent to determining a subset $A \subseteq \mathcal{A}$, whose sum is as close to D as possible.

Second, suppose that $n - \lambda^*$ is even. An optimal schedule for the Lagrangian dual problem is also optimal for the original problem if $W \in [d - p_n, d]$. Finding such a schedule is equivalent to determining a subset $A \subseteq \mathcal{A}$ 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.

Given a subset $A \subseteq \mathcal{A}$ (optimal or approximate), we determine the corresponding schedule for the common due date problem in the following way. Start with $\sigma_{\lambda^*}^{\min}$. Interchange the jobs that correspond to $a_j \in A$ for $j = 1, \ldots, m$, thereby increasing the amount of work processed before d by a_j. Finally, shift the schedule to ensure that the first job is started at time zero.

We show now how to determine a suitable set $A \subseteq \mathcal{A}$. We reindex the values a_j in order of nondecreasing values. For n not too small, the instances of the subset-sum problem virtually always possess the *divisibility* property.

DEFINITION. *A multiset of values* $\{a_1, \ldots, a_m\}$, *with* $1 = a_1 \leqslant a_2 \leqslant \cdots \leqslant a_m$ *is said to possess the divisibility property if for every* j ($j = 1, \ldots, m$) *and for every value* $D \in \{1, 2, \ldots, \Sigma_{i=1}^{j} a_i\}$ *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* $1 = a_1 \leqslant a_2 \leqslant \cdots \leqslant a_m$ *possesses the divisibility property if and only if* $a_{j+1} \leqslant \Sigma_{i=1}^{j} a_i + 1$, *for* $j = 1, \ldots, n-1$. □

An intuitive reason explains why virtually all instances have this property. Every a_j is equal to the difference in processing times between two successive jobs in the shortest processing time order. This implies that for randomly generated instances of the common due date problem the values a_j tend to be small if n is not too small. Note that $\Sigma_{j=1}^{m} a_j \leqslant \max_{1 \leqslant j \leqslant n} p_j$.

THEOREM 7. *If an instance of the subset-sum satisfies the divisibility property, then Johnson's algorithm* (Johnson, 1974), *which is described below, solves this instance to optimality in* $O(m \log m)$ *time.* □

JOHNSON'S ALGORITHM

Step 0. Reindex the values a_j in order of nonincreasing values.

Step 1. Select the largest remaining value a_j with $a_j \leqslant D$. If there is no such value, then stop.

Step 2. Put a_j in the subset; $D \leftarrow D - a_j$.

Step 3. If $D \geqslant a_1$ and if a_1 is not in the subset, then go to Step 1.

Johnson's algorithm always yields a subset whose sum is no more than D. This handicap is overcome by not only applying the algorithm with the value D but also with the value $\Sigma_{j=1}^{m} a_j - D$. If $\bar{A}$ is the subset in the latter case, then an approximate schedule can be constructed as described above by taking $A = \mathcal{A} \setminus \bar{A}$.

If Johnson's algorithm does not yield a provably optimal solution, then we solve the instance to optimality by dynamic programming. This requires $O(mD)$ time and space. By this, we may improve both on the upper and lower bound. If the lower and the upper bound still do not coincide, then we need to apply branch-and-bound to solve the common due date problem to optimality.

The approximation algorithm described above can be adjusted in an obvious fashion to deal with the restricted variant of the common due date problem with $\alpha \neq \beta$.

5. Branch-and-bound

We describe the branch-and-bound algorithm for the case $\alpha = \beta$. The first step in the algorithm is to 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, the jobs must be scheduled in the interval $[0, \Sigma_{j=1}^{n} p_j]$ according to Proposition 1. We then determine upper bounds as described in Section 4; 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 following observation, which is easily verified through an interchange argument. In any optimal schedule, we have that the jobs that are completed before or at the common due date d are scheduled in order of nonincreasing processing times, and that the jobs that are started at or after d are scheduled in order of nondecreasing processing times. In addition, there may be a job that is started before and finished after d. For this particular job, it holds that the early or the tardy jobs have larger processing times. Due to this structure, optimal schedules are said to be V-shaped. Assume now that the jobs have been reindexed 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 $(k = 1, \ldots, n - j)$ descendant, 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 $J_1, \ldots, J_{j+k}$ can easily be computed.

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.

As far as lower bounding in the nodes of the tree is concerned, we only compute the lower bound $L(\lambda^*)$. Hence, we neither solve the modified Lagrangian dual problem nor compute additional upper bounds.

n	t	# $O(n\log n)$	# DP	maximum # nodes	# greedy 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
75	0.1	100	0	0	100	75	100
75	0.2	100	0	0	100	79	100
75	0.3	100	0	0	100	78	100
75	0.4	100	0	0	100	83	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
200	0.1	100	0	0	100	96	100
200	0.2	100	0	0	100	98	100
200	0.3	100	0	0	100	99	100
200	0.4	100	0	0	100	97	100
500	0.1	100	0	0	100	99	100
500	0.2	100	0	0	100	100	100
500	0.3	100	0	0	100	100	100
500	0.4	100	0	0	100	100	100
1000	0.1	100	0	0	100	100	100
1000	0.2	100	0	0	100	100	100
1000	0.3	100	0	0	100	100	100
1000	0.4	100	0	0	100	100	100

TABLE 1. Computational results.

6. Computational results

The processing times were drawn from the uniform distribution [1, 100]. Computational experiments were performed with $d = \lfloor t\Sigma p_j \rfloor$ for $t = 0.1, 0.2, 0.3, 0.4$, respectively, and with the number of jobs ranging from 10 to 1000. 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.

The results are shown in Table 1, the design of which reflects our three-phase approach. The third column '# $O(n\log n)$' shows the number of times (out of 100) that Johnson's subset-sum algorithm gave rise to a schedule with 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 optimality by dynamic programming applied to subset-sum. The fifth column 'maximum # nodes' shows the maximum number of nodes that was needed for the branch-and-bound algorithm. The sixth column '# greedy optimal' shows the number of times that Johnson's algorithm induced 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 (strenghtened or not) was equal to the optimal solution value.

From these results we may draw the conclusion that the common due date problem is extremely easy to solve from a practical point of view. If $n \geqslant 40$, then the $O(n\log n)$ algorithm solves all randomly generated instances to optimality; for $n \geqslant 30$, dynamic programming applied to subset-sum suffices to solve the remaining instances; for $n \leqslant 20$, branch-and-bound is occasionally needed, but requires only a very small number of nodes, and always less than 1 second of running time.

Acknowledgement

The authors would like to thank Jan Karel Lenstra for his helpful comments on an earlier draft of this paper.

References

U. Bagchi, 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 Research Logistics 34*, 739-751.

K. Baker and G. Scudder (1990). Sequencing with earliness and tardiness penalties: a review. *Operations Research 38*, 22-57.

H. Emmons (1987). Scheduling to a common due date on parallel uniform processors. *Naval Research Logistics 34*, 803-810.

N.G. Hall, W. Kubiak, and S.P. Sethi (1990). Deviation of completion times about a restrictive common due date. To appear in *Operations Research*.

J.A. Hoogeveen and S.L. van de Velde (1990). Scheduling around a small common due date. To appear in *European Journal of Operational Research*.

D.S. Johnson (1974). Approximation algorithms for Combinatorial Problems. *Journal of Computer and System Sciences 9*, 256-278.

J.J. KANET (1981). Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly 28*, 643-651.

P.S. SUNDARARAGHAVAN AND M.U. AHMED (1984). Minimizing the sum of absolute lateness in single-machine and multimachine scheduling. *Naval Research Logistics Quarterly 31*, 325-333.

W. SZWARC (1989). Single machine scheduling to minimize absolute deviation of completion times from a common due date. *Naval Research Logistics 36*, 663-673.