Minimizing total inventory cost on a single machine in just-in-time manufacturing

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computing Science

Memorandum COSOR 92-20

Minimizing Total Inventory Cost on a Single Machine in Just-in-Time Manufacturing

J.A. Hoogeveen
S.L. van de Velde

Eindhoven, June 1992
The Netherlands

Eindhoven University of Technology
Department of Mathematics and Computing Science
Probability theory, statistics, operations research and systems theory
P.O. Box 513
5600 MB Eindhoven - The Netherlands

Secretariate: Dommelbuilding 0.03
Telephone: 040-47 3130

ISSN 0926 4493

Minimizing Total Inventory Cost on a Single Machine in Just-in-Time Manufacturing

J.A. Hoogeveen
Department of Mathematics and Computing Science,
Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

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

The just-in-time concept decrees not to accept ordered goods before their due dates in order to avoid inventory cost. This bounces the inventory cost back to the manufacturer: products that are completed before their due dates have to be stored. Reducing this type of storage cost by preclusion of early completion conflicts with the traditional policy of keeping work-in-process inventories down. This paper addresses a single-machine scheduling problem with the objective of minimizing total inventory cost, comprising cost associated with work-in-process inventories and storage cost as a result of early completion. The cost components are measured by the sum of the job completion times and the sum of the job earlinesses. This problem differs from more traditional scheduling problems, since the insertion of machine idle time may reduce total cost. The search for an optimal schedule, however, can be limited to the set of job sequences, since for any sequence there is a clear-cut way to insert machine idle time in order to minimize total inventory cost. We apply branch-and-bound to identify an optimal schedule. We present five approaches for lower bound calculation, based upon relaxation of the objective function, of the state space, and upon Lagrangian relaxation.

1980 Mathematics Subject Classification (1985): 90B35.
Key Words and Phrases: just-in-time manufacturing, inventory cost, work-in-process inventory, earliness, tardiness, machine idle time, branch-and-bound algorithm, Lagrangian relaxation.

1. Introduction

The just-in-time concept has affected the attitude towards inventories significantly. In order to keep inventories down, there is a reluctance to accept ordered goods prior to their due dates. This implies that manufacturers have to store early completed goods before they can be shipped to their destinations. This has added a relatively new aspect to machine scheduling theory: the preclusion of earliness. In principle, earliness can be avoided by allowing machine idle time, thereby deferring jobs. Machine idleness, however, runs counter to the natural instinct to minimize work-in-process inventories, to maximize machine utilization, and to observe due dates.

Within this context, we address the following situation. A set $\mathcal{J}=\{J_1,\ldots,J_n\}$ of n independent jobs has to be scheduled on a single machine, which is continuously available from time zero onwards. The machine can handle at most one job at a time. Job J_j $(j=1,\ldots,n)$ requires a positive integral uninterrupted processing time p_j and should ideally be completed exactly on its due date d_j. A *schedule* specifies for each job J_j a completion time C_j such that the jobs do not overlap in their execution. The order in which the machine processes the jobs is called the *job sequence*. For a given schedule, the earliness of J_j is defined as $E_j=\max\{0, d_j-C_j\}$ and its tardiness as $T_j=\max\{0, C_j-d_j\}$. In addition, we define *maximum earliness* as $E_{\max}=\max_{1\le j\le n} E_j$ and *maximum tardiness* as $T_{\max}=\max_{1\le j\le n} T_j$. Accordingly, J_j is called *early*, *just-in-time*, or *tardy* if $C_j<d_j$, $C_j=d_j$, or $C_j>d_j$, respectively.

In this paper, we follow the terminology of Graham, Lawler, Lenstra, and Rinnooy Kan (1979) to classify scheduling problems. Deterministic scheduling problems are classified according to a three-field notation $\alpha\,|\,\beta\,|\,\gamma$, where α specifies the machine environment, β the job characteristics, and γ the objective function. For instance, $\alpha=1$ refers to a single machine, $\beta=pmtn$ signifies that the jobs may be preempted, that is, the processing of a job may be interrupted and resumed later, and $\gamma=\Sigma C_j$ means that the objective is to minimize the sum of the job completion times. Since earliness is nonincreasing in the job completion times, it may generally be advantageous to permit machine idle time. The inclusion of the acronym *nmit* in the second field signifies that no machine idle time is allowed.

Three types of single-machine scheduling problems involving job earliness have been considered in the literature. The best-known is the minimization of $E_{\max}$. If machine idle time is not allowed, then the problem is solved by scheduling the jobs in nondecreasing order of d_j-p_j; this is known as the *minimum slack time order*. If machine idle time is permitted, then the problem is trivial: for any given sequence, we defer the jobs until all are just-in-time or tardy. This approach also applies to $1\,|\,|\,\Sigma E_j$, but, surprisingly, $1\,|\,nmit\,|\,\Sigma E_j$ is $\mathcal{NP}$-hard in the ordinary sense (Du and Leung, 1990). The third problem is to maximize $\Sigma w_j E_j$, where w_j is the weight of job J_j, denoted as $1\,|\,|\,-\Sigma w_j E_j$; this problem is solvable in pseudopolynomial time by an algorithm due to Lawler and Moore (1969).

The combination of earliness with another performance measure, reflecting other considerations, takes us into the arena of bicriteria scheduling. The state of the art, as far as a measure of earliness is concerned, is as follows. For the $1\,|\,pmtn,\ nmit\,|\,\alpha\Sigma C_j+\beta E_{\max}$ problem, Hoogeveen and Van de Velde (1990) present an algorithm that runs in $O(n^4)$ time. They show that the same algorithm also solves $1\,|\,|\,\alpha\Sigma C_j+\beta E_{\max}$ in case $\alpha\ge\beta$. Hoogeveen (1990) presents algorithms that solve $1\,|\,|\,\alpha E_{\max}+\beta T_{\max}$ and $1\,|\,nmit\,|\,F(E_{\max},T_{\max})$ in $O(n^2\log n)$ and $O(n^2)$ time; F is here an arbitrary nondecreasing function of $E_{\max}$ and $T_{\max}$. For the $1\,|\,nmit\,|\,\Sigma(\alpha_j E_j+\beta T_j)$ problem, Ow and Morton (1989) propose a local search method to generate approximate solutions. A voluminous part of research is concerned with common due date scheduling. Here, we have $d_j=d$ $(j=1,\ldots,n)$; the

objective is to minimize some function of earliness and tardiness. A survey of problems, algorithms, and computational complexity is provided by Baker and Scudder (1990).

In this paper, we consider the problem of minimizing total inventory cost, which is supposed to comprise two components: cost due to work-in-process inventory and storage cost as a result of early completions. These components are assumed to depend linearly on the sum of job completion times and the sum of job earliness. If we let α and β denote the cost per unit time for work-in-process inventory and storage of finished product, respectively, then the total inventory cost for a given schedule σ is

$$f(\sigma)=\alpha\sum_{j=1}^{n}C_j+\beta\sum_{j=1}^{n}E_j.$$

Without loss of generality, we assume α and β to be integral, positive, and relatively prime. Since we have by definition that $E_j=T_j-C_j+d_j$ for $j=1,\ldots,n$, the objective function can alternatively be written as

$$(\alpha-\beta)\sum_{j=1}^{n}C_j+\beta\sum_{j=1}^{n}(T_j+d_j).$$

If $\alpha\geq\beta$, then this a regular objective function, and hence there is an optimal schedule without machine idle time. The case $\alpha=\beta$ reduces to $1\,||\,\Sigma T_j$, which is $\mathcal{NP}$-hard in the ordinary sense (Du and Leung, 1990). Garey, Tarjan, and Wilfong (1988) prove that the case $\alpha<\beta$ is $\mathcal{NP}$-hard, too. We note that the case $\beta>n\alpha$ reduces to $1\,|\,r_j\,|\,\Sigma C_j$, which is also $\mathcal{NP}$-hard in the strong sense (Lenstra, Rinnooy Kan, and Brucker, 1977).

We address the case $\beta\geq\alpha$, in which the insertion of machine idle time may be advantageous. Our purpose is to find a feasible schedule σ that minimizes $f(\sigma)$. This problem was introduced by Fry and Keong Leong (1987a), who formulate it as an integer linear program. They used a standard code to find an optimal schedule. Not surprisingly, the proposed method solves problems up to 12 jobs only.

The search for an optimal schedule, however, can be reduced to a search over the $n!$ different job sequences, as there is a clear-cut method to insert machine idle time to minimize total cost for a *given* sequence. This method, which requires $O(n^2)$ time, is described in Section 2.

The freedom to leave the machine idle singles out our problem from most concurrent research on scheduling problems with earliness penalties. To our knowledge, this is the first paper that presents a branch-and-bound algorithm for a single-machine scheduling problem with a nonregular objective function, where insertion of machine idle time is allowed. Machine idle time affects the design of a branch-and-bound algorithm significantly. In Section 3, we discuss some components of the algorithm such as the upper bound, the branching rule, the search strategy, and the dominance rules. Lower bounds are presented in Section 4. The range of the due dates in proportion to the processing times mainly dictates when the first job is started and how much machine idle time is inserted between the execution of the jobs. To cope with the variety of due date patterns, we propose five approaches for lower bound computation. Each of these methods seems to be suitable for a certain class of instances. Some computational results are reported in Section 5; conclusions are presented in Section 6.

2. The insertion of idle time for a given sequence

The search for an optimal schedule can be reduced to a search over the $n!$ different job sequences, as there is a clear-cut procedure to insert machine idle time so as to minimize total cost for a *given* sequence.

This procedure, however, is not new. Similar methods have been presented (cf. Baker and Scudder, 1990), including the ones proposed by Fry and Keong Leong (1987b) for the $1||\Sigma(\alpha C_j+\beta E_j+\gamma T_j)$ problem and by Garey, Tarjan, and Wilfong (1988) for the $1||\Sigma(E_j+T_j)$ problem. This is not surprising: as we have already noted, $T_j=C_j+E_j-d_j$ for all j; for specific choices for α and β, our problem is equivalent with theirs.

Suppose that the scheduling order is $\sigma=(J_n,\ldots,J_1)$. Accordingly, $\bar{C}_j=\Sigma_{k=j}^{n} p_k$ is the earliest possible completion time of J_j in this sequence. We introduce a vector $x=(x_1,\ldots,x_n)$ of variables, with x_j $(j=1,\ldots,n)$ denoting the amount of idle time immediately before the execution of J_j. The actual completion time of J_j is then $C_j=\bar{C}_j+\Sigma_{k=j}^{n}x_k$. The problem of minimizing inventory cost for the given job sequence is then equivalent to determining values x_j $(j=1,\ldots,n)$ that minimize

$$\alpha\sum_{j=1}^{n}(\bar{C}_j+\sum_{k=j}^{n}x_k)+\beta\sum_{j=1}^{n}\max(0,\, d_j-\bar{C}_j-\sum_{k=j}^{n}x_k)$$

subject to

$$x_j\geq 0, \qquad \text{for } j=1,\ldots,n.$$

By the introduction of auxiliary variables E_j denoting the earliness of J_j $(j=1,\ldots,n)$, we can easily transform this problem into a linear programming problem. We know therefore that the optimum is attained in a vertex of the unspecified LP polytope. In addition, we know that the optimal x_j are integral, since the due dates, the processing times, α, and β are integral. A necessary condition for x to be optimal is that all existing *primitive directional derivatives* at x are non-negative. The primitive directional deratives are equal to the change of the scheduling cost if x_j is increased by one unit, and the change of the scheduling cost if x_j is decreased by one unit, for $j=1,\ldots,n$. The increase of x_j by one unit only affects J_j and the jobs succeeding J_j up to the first period of machine idle time after J_j. We call these jobs the *immediate successors* of J_j. Let Q_j denote the set containing J_j and its immediate successors, let n_j be the number of early jobs in Q_j, and let g_j be the primitive directional derivative for increasing x_j. We have then that $g_j=\alpha\,|Q_j|-\beta\,n_j$. Recall that each J_j is ideally completed on its due date d_j.

Using the above observations, we develop an inductive procedure for finding an optimal schedule for σ. This procedure finds an optimal schedule for the subsequence $(J_l,\ldots,J_1)$, given an optimal schedule for the subsequence $(J_{l-1},\ldots,J_1)$, for $l=2,\ldots,n$. The first step is to find out whether putting $C_l=d_l$ is feasible; if so, then we have an optimal schedule for $(J_l,\ldots,J_1)$. Suppose $C_l=d_l$ is not feasible, because J_l overlaps with some other job. We then tentatively put $C_l=C_{l-1}-p_{l-1}$, and compute the optimal deferral of the jobs in Q_l, disregarding the jobs not in Q_l. The optimal deferral, denoted by δ, is dictated by the first point where g_l becomes non-negative. This deferral is feasible if δ is no larger than the length of the period of idle time immediately after the last job in Q_l; let this length be $\delta_{\max}$. If $\delta\leq\delta_{\max}$, then we get an optimal schedule for $(J_l,\ldots,J_1)$ by deferring the jobs in Q_l by δ. If $\delta>\delta_{\max}$, then we defer the jobs in Q_l by $\delta_{\max}$. At this point, we repeat the process for J_l: we update Q_l, and evaluate if additional deferral of the jobs in Q_l is advantageous. We now give a step-wise description of the idle time insertion algorithm.

Idle time insertion algorithm

Step 0. $C_1 \leftarrow d_1$; $l \leftarrow 2$.

Step 1. If $l = n+1$, go to Step 9.

Step 2. Put $C_l \leftarrow \min\{d_l, C_{l-1} - p_{l-1}\}$. If $C_l = d_l$, then go to Step 8.

Step 3. Determine Q_l and evaluate g_l. If $g_l \geq 0$, then go to Step 8.

Step 4. Compute E_j for each job $J_j \in Q_l$.

Step 5. Compute δ_{max}, i.e., the length of the period of idle time immediately after the last job in Q_l.

Step 6. Let $a \leftarrow \lfloor (|Q_l|)\alpha/\beta \rfloor$, and $k \leftarrow |Q_l| - a$. Determine the kth smallest value of the earlinesses of the jobs in Q_l; this value is denoted as $E_{[k]}$. If the jobs in Q_l are deferred by $\delta = E_{[k]}$, then at most a jobs in Q_l remain early; due to the choice of a, g_l then becomes non-negative.

Step 7. Defer the jobs in Q_l by $\Delta = \min\{\delta, \delta_{max}\}$. If $\delta > \delta_{max}$, then go to Step 3.

Step 8. $l \leftarrow l+1$; go to Step 1.

Step 9. An optimal schedule for the sequence $(J_n, \ldots, J_1)$ has been determined.

Theorem 1. *The idle time insertion algorithm generates an optimal schedule for a given sequence.*

Proof. The proof proceeds by induction. The algorithm clearly produces the optimal schedule in case of a single job. Suppose that we want to find an optimal schedule for the sequence $(J_l, \ldots, J_1)$, having an optimal schedule for the sequence $(J_{l-1}, \ldots, J_1)$ available. There are two cases to consider. First, suppose $d_l \leq C_{l-1} - p_{l-1}$; in this case, we let $C_l = d_l$, and retain the completion times of the other jobs; this specifies an optimal schedule for the sequence $(J_l, \ldots, J_1)$. Suppose now $d_l > C_l - p_l$; for this case, deferring J_{l-1} and thereby its immediate successors, i.e., the jobs contained in the set Q_{l-1}, may be advantageous. We can compute the cost of deferring Q_{l-1} by one unit; the benefit of deferring J_l by one unit is equal to $\beta - \alpha$. If the cost is higher than or equal to the benefit, then we put $C_l = C_{l-1} - p_{l-1}$, and we have an optimal schedule for $(J_l, \ldots, J_1)$; otherwise, we defer the jobs in Q_{l-1} by one unit, and evaluate whether additional deferral is advantageous. The idle time insertion algorithm shortcuts this procedure by computing the break-even point, that is, the point where additional deferral is not advantageous. □

Consider the example for which the data are given in Table 1. Let $\alpha = 1$ and $\beta = 4$. We construct the optimal schedule for the sequence (J_3, J_2, J_1). First, we put $C_1 = d_1 = 15$. Next, we let $C_2 = d_2 = 10$, as $d_2 \leq C_1 - p_1$. Note that $d_3 > C_2 - p_2$. Therefore, we tentatively put $C_3 = C_2 - p_2 = 7$, and consider deferring J_3 and J_2. Apparently, we have $Q_3 = \{J_3, J_2\}$, $n_3 = 1$, $g_3 = 2\alpha - \beta < 0$, and $E_{[2]} = 3$. However, $\delta_{max} = C_1 - p_1 - C_2 = 2$, therefore, we defer J_2 and J_3 by 2 units. At this point, the three jobs are processed consecutively. Now we have $g_3 = 3\alpha - \beta$, and additional deferral is still advantageous. As $E_{[3]} = 1$, we insert one more unit of machine idle time. The optimal schedule for each subproblem is depicted in Figure 1.

The algorithm runs in $O(n^2)$ time. A complete run through the main part of the algorithm, i.e., steps 2 through 8, takes $O(n)$ time: this is needed to identify the set Q_l, to compute the primitive directional derivative g_l, the values δ_{max} and δ, and to defer the jobs, if necessary. The value δ is determined in $O(n)$ time through a median-finding technique; see Aho, Hopcroft, and Ullman (1982). After each run through the main part of the algorithm, a gap between two successive jobs is closed. As at most $n-2$ such gaps exist, the algorithm runs in $O(n^2)$ time. For the case $2\alpha = \beta$, i.e., for the problem

J_j	p_j	d_j
J_1	3	15
J_2	3	10
J_3	6	10

Table 1. Data for the example.

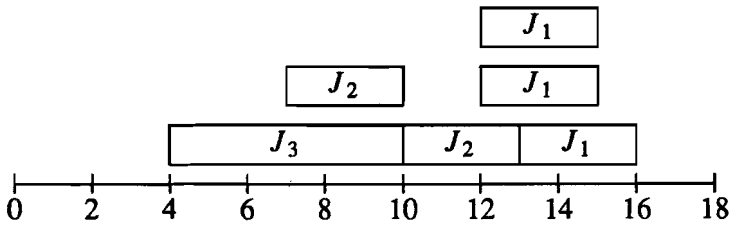

Figure 1. Schedules for the example.

$1||\Sigma(E_j+T_j)$, Garey, Tarjan, and Wilfong (1988) show that the idle time insertion procedure can be implemented to run in $O(n\log n)$ time.

The problem of inserting machine idle time can also be solved by a symmetric procedure starting with the first job in σ. Because of our specific branching rule, however, we choose to start at the end.

In the remainder, we use the terms sequence and schedule interchangeably. Unless stated otherwise, σ also refers to the optimal schedule for the sequence σ and to the set of jobs in the sequence σ. From now on, we let $p(\sigma)=\Sigma_{J_j\in\sigma} p_j$.

3. The branch-and-bound algorithm

We adopt a *backward sequencing branching rule*: a node at level k of the search tree corresponds to a sequence π with k jobs fixed in the last k positions. We assume from now on that the first job in a partial schedule π is not started before time $p(\mathcal{J}-\pi)$; this additional restriction, imposed to leave space for the remaining jobs, is easily incorporated in the idle time insertion algorithm. Let $f(\pi)$ denote the minimal inventory cost for π. Let $\bar{f}(\pi)$ denote the minimal inventory cost for π if the first job may be started before time $p(\mathcal{J}-\pi)$; the notation $\bar{f}(\pi)$ is only needed in this section. For any partial schedule π, we have $f(\pi)\geq\bar{f}(\pi)$.

We employ a *depth-first* strategy to explore the tree: at each level, we generate the descendant nodes for only one node at a time. At level k, there are $n-k$ descendant nodes: one for each unscheduled job. The completion times for the jobs in π are only temporary. Branching from a node that corresponds to π, we add some job J_j leading to the sequence $J_j\pi$. Subsequently, we determine the associated optimal schedule for $J_j\pi$, and possibly defer some jobs in π. We branch from the nodes in order of non-increasing due dates of the associated jobs. Before entering the search tree, we determine an upper bound on the optimal solution value. We use the optimal schedule corresponding to the minimum slack time sequence as an initial solution, and try to reduce its cost by pairwise adjacent interchanges.

A node is discarded if its associated partial schedule π cannot lead to a complete schedule with cost less than UB; UB denotes the currently best solution value. Let $LB(\mathcal{J}-\pi)$ be some lower bound on the minimal cost of scheduling the jobs in the set $\mathcal{J}-\pi$. Obviously, we discard a node if

$f(\pi)+LB(\mathcal{J}-\pi) \geq UB$. The following rule is usually overlooked. Let $g(\sigma_1,\sigma_2)$ be a lower bound on the cost for scheduling the jobs in σ_1 given the final partial schedule σ_2.

Theorem 2. *The partial schedule π can be discarded if there exists a $J_j \in \mathcal{J}-\pi$ for which $\bar{f}(J_j\pi)+g(\mathcal{J}-\pi-J_j,\pi) \geq UB$.*

Proof. Consider a complete sequence σ that has π as final subsequence. Thus, σ can be written as $\sigma=\pi_1 J_j \pi_2 \pi$. Accordingly, we have

$$f(\sigma)=f(\pi_1 J_j \pi_2 \pi) \geq \bar{f}(J_j\pi)+g(\pi_1\pi_2,\pi) \geq UB. \quad \square$$

It is essential that $g(\mathcal{J}-\pi-J_j,\pi)$ depends only on π and not on $J_j\pi$, and that we use $\bar{f}(J_j\pi)$ instead of $f(J_j\pi)$. We derive two corollaries from Theorem 2.

Corollary 1. *If for a given partial schedule π, we have that $\bar{f}(J_jJ_k\pi)+g(\mathcal{J}-\pi-J_j-J_k,\pi) \geq UB$ for some $J_j \in \mathcal{J}-\pi$ and $J_k \in \mathcal{J}-\pi$, then J_k precedes J_j in any complete schedule $\sigma\pi$ with $f(\sigma\pi) < UB$.* $\square$

Corollary 2. *The partial schedule π can be discarded if two jobs $J_j \in \mathcal{J}-\pi$ and $J_k \in \mathcal{J}-\pi$ exist with $g(\mathcal{J}-\pi-J_j-J_k,\pi)+\min\{\bar{f}(J_jJ_k\pi),\bar{f}(J_kJ_j\pi)\} \geq UB$.* $\square$

If a partial schedule $\pi^* \neq \pi$ exists comprising the same jobs as π and having $f(\sigma\pi^*) \leq f(\sigma\pi)$ for any sequence σ for the remaining $n-k$ jobs, then we can also discard π. If $f(\sigma\pi^*) < f(\sigma\pi)$ for some σ, then π is *dominated* by π^*. If $f(\sigma\pi^*)=f(\sigma\pi)$ for every σ, then we discard either π^* or π. The dominance condition above can be narrowed by the requirement that $f(\pi^*) \leq f(\pi)$ and that the circumstances to add the remaining $n-k$ jobs to π^* are at least as good as the circumstances to add the remaining jobs to π. The question whether such a sequence π^* exists is of course $\mathcal{NP}$-complete. We strive therefore to identify sufficient conditions to discard π. The temporary nature of the job completion times for π complicates the achievement of this goal. We have to be careful with dominance conditions that are based on interchange arguments: the conditions must remain valid if the jobs in π are deferred.

Suppose that the jobs in π have been reindexed in order of increasing completion times. In each of the following theorems, stating the dominance rules, the sequence π^* is obtained from π by swapping two jobs, say, J_j and J_k. We do not compute the optimal completion times for the sequence π^*. Instead, we determine the job completion times for the sequence π^* as follows. Let C_i and C_i^* be the completion time of J_i in the schedule π and π^*, respectively. Then we let

$$\begin{aligned}
&C_i^* = C_i, && \text{for } i=1,\ldots,j-1, i=k+1,\ldots,|\pi|,\\
&C_i^* = C_i-p_j+p_k, && \text{for } i=j+1,\ldots,k-1,\\
&C_k^* = C_j-p_j+p_k,\\
&C_j^* = C_k.
\end{aligned}$$

Let $F(\pi^*)$ be the cost associated with the completion times C_i^*, for $i=1,\ldots,|\pi|$. Hence, $F(\pi^*) \geq f(\pi^*)$. To validate the following dominance rules, we must verify that $f(\pi) \geq F(\pi^*)$, even if the jobs are deferred. Due to the relation between π and π^*, this comes down to verifying that for each set of nonnegative values Δ_i $(i=1,\ldots,n)$

$$\alpha\sum_{i=j}^{k} C_i+\beta\sum_{i=j}^{k}\max\{0,d_i-C_i-\Delta_i\}\geq\alpha\sum_{i=j}^{k} C_i^*+\beta\sum_{i=j}^{k}\max\{0,d_i-C_i^*-\Delta_i\}. \quad (1)$$

We start with a straightforward result.

Theorem 3. *There is an optimal schedule with J_j preceding J_k if $p_j=p_k$ and $d_j\leq d_k$.* □

Theorem 4. *The partial sequence π can be discarded if there are two jobs J_j and J_k with $C_k=C_j+\Sigma_{i=j+1}^{k} p_i$ for which*

$$p_j>p_k, \text{ and}$$

$$\alpha\sum_{i=j}^{k} C_i+\beta\sum_{i=j+1}^{k}\max\{0,d_i-C_i\}\alpha\sum_{i=j}^{k} C_i^*+\beta\sum_{i=j+1}^{k}\max\{0,d_i-C_i^*\}. \quad (2)$$

Proof. As there is no idle time between the jobs in the block that begins with J_j and ends with J_k, the idle time insertion algorithm will defer all jobs in this block by the same amount of time Δ. Define $c(\Delta)$ as the change of cost due to the interchange, after deferring the jobs by $\Delta\geq 0$; i.e.,

$$c(\Delta)=\alpha\sum_{i=j}^{k} C_i+\beta\sum_{i=j}^{k}\max\{0,d_i-C_i-\Delta\}-\alpha\sum_{i=j}^{k} C_i^*-\beta\sum_{i=j}^{k}\max\{0,d_i-C_i^*-\Delta\}.$$

We prove that $c(\Delta)\geq 0$ for all $\Delta\geq 0$. From condition (2), it follows immediately that $c(0)\geq 0$. Furthermore, $C_j<C_j^*$ implies $\max\{0,d_j-C_j-\Delta\}\geq\max\{0,d_j-C_j^*-\Delta\}$ for all $\Delta\geq 0$; $C_i>C_i^*$ for $i=j+1,\ldots,k$ implies $\max\{0,d_i-C_i-\Delta\}-\max\{0,d_i-C_i^*-\Delta\}\geq\max\{0,d_i-C_i\}-\max\{0,d_i-C_i^*\}$ for all $\Delta\geq 0$. Combining the inequalities, we get the desired result. □

The possible increase of E_j is excluded here. The following theorem shows that in case no idle time exists between two adjacent jobs, then dominance already exists if condition (1) is satisfied for $\Delta=0$.

Theorem 5. *The partial sequence π can be discarded if there are two jobs J_j and J_k with $C_k=C_j+p_k$ for which*

$$p_j>p_k,$$

and

$$\alpha(p_j-p_k)+\beta\max\{0,d_j-C_j\}+\beta\max\{0,d_k-C_k\}\geq$$
$$\beta\max\{0,d_j-C_k\}+\beta\max\{0,d_k-C_k+p_j\}. \quad (3)$$

Proof. Define $c(\Delta)$ as the change of cost due to the interchange, after deferring the jobs by $\Delta\geq 0$; i.e.,

$$c(\Delta)=\alpha(p_j-p_k)+\beta\max\{0,d_j-C_j-\Delta\}-\beta\max\{0,d_j-C_k-\Delta\}+$$
$$\beta\max\{0,d_k-C_k-\Delta\}-\beta\max\{0,d_k-C_k+p_j-\Delta\}.$$

We need to show that condition (3), stating that $c(0) > 0$, implies $c(\Delta) \geq 0$ for all $\Delta \geq 0$. Note that $\alpha < \beta$ implies that at least one due date is smaller than C_k; otherwise, condition (3) is not valid.

The expression $c(\Delta)$ has three components. The first component is $\alpha(p_j - p_k)$; it is a constant. The second component is $\beta \max\{0, d_j - C_j - \Delta\} - \beta \max\{0, d_j - C_k - \Delta\}$; it is a piecewise linear function of Δ. The function value is βp_k if $d_j \geq C_k + \Delta$, and 0 if $d_j \leq C_j + \Delta$. If $C_k + \Delta > d_j \geq C_j + \Delta$, then the gradient is -1. The third component is $\beta \max\{0, d_k - C_k - \Delta\} - \beta \max\{0, d_k - C_k + p_j - \Delta\}$; it is also a piecewise linear function of Δ. The function value is $-\beta p_j$ if $d_k \geq C_k + \Delta$, and 0 for $d_k \leq C_k - p_j + \Delta$. The gradient is 1 if $C_k + \Delta > d_k \geq C_k - p_j + \Delta$. Combining the three components yields a piecewise linear function whose behavior depends on the due dates. We now make the following observations. First, $c(\Delta) > 0$ if $\Delta \geq d_k - C_k + p_j$. Second, if $c(t) > 0$ for some $t \geq d_k - C_k$, then $c(\Delta) > 0$ for all $\Delta \geq t$. As at least one due date is smaller than C_k, the second observation implies that, if $d_k \leq d_j$, then $c(\Delta) > 0$ for all $\Delta \geq 0$.

The only case left to consider is $d_j < d_k$ and $0 \leq \Delta \leq d_k - C_k$. Then, we have $c(\Delta) = \alpha(p_j - p_k) - \beta p_j + \beta \max\{0, d_j - C_j - \Delta\}$. As $d_j - C_j - \Delta \leq d_j - C_j = d_j - C_k + p_k \leq p_k$, we get $c(0) \leq (\alpha - \beta)(p_j - p_k) \leq 0$, which contradicts the assumption. This completes the proof. □

In Corollary 3, explicit conditions for the existence of dominance are derived from Theorem 5. This corollary is referred to when lower bounds are discussed in Section 4.

Corollary 3. *The partial sequence π can be discarded if there are two jobs J_j and J_k with $C_k = C_j + p_k$ such that*

$$p_j > p_k,$$

and one of the following conditions is satisfied:

$$C_k - p_j \geq d_k,$$

$$C_k - p_j < d_k,\ C_k \geq d_k,\ \alpha(p_j - p_k) \geq \beta(d_k - C_k + p_j),$$

$$C_k - p_j < d_k,\ C_k < d_k,\ \alpha(p_j - p_k) \geq \beta p_j,$$

$$C_k - p_j < d_k,\ C_k \geq d_k,\ \alpha(p_j - p_k) \geq \beta(d_k - d_j - p_k + p_j).$$ □ □

Theorem 6. *The partial sequence π with J_k scheduled last is dominated if there is a J_j such that*

$$p_j > p_k, \text{ and } C_j - p_j + p_k \geq d_k.$$

Proof. Let $\pi = \pi_1 J_j \pi_2 J_k$ and $\pi^* = \pi_1 J_k \pi_2 J_j$. We compute the effect of the interchange on the scheduling cost. Since J_k is the last job in the optimal schedule π, we have $C_k \geq d_k$. In addition, we know $C_j^* = \max\{d_j, C_k - p_k + p_j\}$ and $C_k^* = C_j - p_j + p_k \geq d_k$. First, suppose $C_j^* = d_j$. The effect of the interchange is then equal to

$$\alpha(C_j + C_k - (C_j - p_j + p_k) - d_j) + \beta(d_j - C_j) \geq \alpha(C_k + p_j - p_k - d_j) + \alpha(d_j - C_j) > 0,$$

as $C_k - p_k \geq C_j$. Second, suppose that $C_j^* = C_k - p_k + p_j$. The effect of the interchange is then equal to

$$\alpha[C_j + C_k - (C_k - p_k + p_j) - (C_j - p_j + p_k)] + \beta \max\{0, d_j - C_j\} \geq 0.$$

The effect remains non-negative if the jobs are deferred. □

Theorem 7. *There is an optimal schedule in which J_k is not scheduled in the last position, if there is some J_j with $p_j > p_k$ and $d_j - p_j \geq d_k - p_k$.*

Proof. We let $\pi = \pi_1 J_j \pi_2 J_k$ and $\pi^* = \pi_1 J_k \pi_2 J_j$ and compute the effect of the interchange. We have $C_k \geq d_k$ and $C_k - p_k \geq C_j$; in addition, we define here $C_j^* = \max\{d_j, C_k - p_k + p_j\}$. The effect of the interchange has to be non-negative; we therefore have to prove that

$$\alpha C_k + \beta \max\{0, d_j - C_j\} \geq \alpha(p_k - p_j + C_j^*) + \beta \max\{0, d_k - p_k + p_j - C_j\}. \tag{4}$$

First, we examine the case $C_j^* = C_k - p_k + p_j$. Expression (4) is then equivalent to

$$\beta \max\{0, d_j - C_j\} \geq \beta \max\{0, d_k - p_k + p_j - C_j\},$$

which is true for any C_j since $d_j - p_j \geq d_k - p_k$. Second, consider the case $C_j^* = d_j$. This implies $d_j > C_j$, since $d_j \geq C_k - p_k + p_j > C_j - p_k + p_j > C_j$. Hence, expression (4) is equivalent to

$$\alpha C_k + \beta(d_j - C_j) \geq \alpha(p_k - p_j + d_j) + \beta \max\{0, d_k - p_k + p_j - C_j\}.$$

Suppose $\max\{0, d_k - p_k + p_j - C_j\} = d_k - p_k + p_j - C_j$. We must then verify that

$$\alpha C_k + \beta d_j \geq \alpha(d_j - p_j + p_k) + \beta(d_k - p_k + p_j).$$

As $C_k \geq d_k$, we only need to prove that

$$0 \geq (\alpha - \beta)[(d_j - p_j) - (d_k - p_k)];$$

this expression is true since $\beta > \alpha$ and $d_j - p_j \geq d_k - p_k$. Conversely, suppose $\max\{0, d_k - p_k + p_j - C_j\} = 0$. Since $\alpha C_k + \beta(d_j - C_j) \geq \alpha(C_k + d_j - C_j) \geq \alpha(p_k + d_j) > \alpha(p_k - p_j + d_j)$, expression (4) is also true for this case. □

Corollary 4. *There is an optimal schedule in which J_j is scheduled last if $p_j \geq p_k$ and $d_j - p_j \geq d_k - p_k$ for each $J_k \in \mathcal{J}$.* □

4. Lower bounds

In this section, we present five lower bound procedures. It seems to be impossible to develop a lower bound procedure that copes satisfactorily with all conceivable due date patterns. For example, imagine an instance with due dates small with respect to the sum of the processing times; little idle time needs then to be inserted. In contrast, consider an instance with $d_k \; \Sigma_{j=1}^{n} p_j$ for each J_k; the machine will then be idle for some time before processing the first job. Numerous variations and combinations of both patterns are possible.

Each of the lower bound methods is effective for a specific class of instances. Nonetheless, we use them supplementary rather than complementary. We partition the job set $\mathcal{J}$ into subsets, apply each lower bound method to each subset, and aggregate the best lower bounds. In this way, we hope to obtain a lower bound that is stronger than the separate lower bounds obtained for the entire set $\mathcal{J}$. The success of this strategy depends on the partitioning strategy. The jobs in a subset should be conflicting, that is, they should overlap when completed at their due date. If they are not, then we get the weak lower bound $\alpha\Sigma_{j=1}^{n} d_j$. In this sense, we prefer subsets such that the executions of the jobs in the same subset interfere with each other, but not with the execution of the jobs in the other subsets. We propose two partitioning strategies that pursue this effect.

The first strategy is motivated by the structure of any optimal schedule. The jobs that are consecutively processed between two periods of idle time interfere with each other, but not with the other jobs. Such a partitioning is hard to obtain. To mimic such a partitioning, we identify clusters. A *cluster* is a set of jobs such that for each job J_j in the cluster there is another job J_k in the cluster such that the intervals $[d_j-p_j, d_j]$ and $[d_k-p_k, d_k]$ overlap; hence, for each job in the cluster there exists a conflict with at least one other job in the cluster. However, clusters may interfere with each other in any optimal schedule.

The second strategy is the following. Given a partial schedule π, we try to identify the jobs not in π that will be early in any optimal complete schedule of the form $\sigma\pi$. We call these jobs *surely* early. The idea is to derive an upper bound T on the completion times of the unscheduled jobs; accordingly, $J_j \in \mathcal{J}-\pi$ is surely early if $d_j > T$. For instance, let g be the primitive directional derivative for deferring the first job in π by one unit. Suppose that $|\mathcal{J}-\pi|(\beta-\alpha) \leq g$. The current set of completion times for the jobs in π is then optimal for any schedule $\sigma\pi$; an upper bound T is then the start time of the first job in π. Other upper bounds are derived from the dominance rules. Suppose J_j and J_k are adjacent in π with $p_j > p_k$ and J_j preceding J_k. (It is not necessary that $C_k = C_j + p_k$.) The first condition of Corollary 3 indicates that π is dominated if $C_k \geq d_k + p_j$; hence, an upper bound is given by $d_k + p_j - 1 - \Sigma_{J_i \in \pi,\, C_i \leq C_k} p_i$. From the other criteria in Corollary 3 and from Theorem 7, similar upper bounds are derived. They can also be derived from Theorem 4, but this requires an intricate procedure. Finally, we set T equal to the minimum of all upper bounds. If no upper bound is specified, then we let $T = \infty$.

4.1. First method: relax the objective function

Let $\mathcal{E}$ denote the set of surely early jobs; let $\mathcal{R}$ be the set of remaining jobs. Observe that

$$\min_{\sigma \in \Omega} f(\sigma) \geq \min_{\sigma \in \Omega_{\mathcal{R}}} \sum_{J_j \in \mathcal{R}} \alpha C_j + \min_{\sigma \in \Omega_{\mathcal{E}}} \sum_{J_j \in \mathcal{E}} [\alpha C_j + \beta E_j],$$

where $\Omega_{\mathcal{R}}$ and $\Omega_{\mathcal{E}}$ denote the set of feasible schedules for the jobs in $\mathcal{R}$ and $\mathcal{E}$. The problem of minimizing $\Sigma_{J_j \in \mathcal{E}} [\alpha C_j + \beta E_j]$ is solvable in polynomial time; we have $E_j = d_j - C_j$ for each $J_j \in \mathcal{E}$, and hence, the scheduling cost reduces to $\Sigma_{J_j \in \mathcal{E}} [(\alpha-\beta)C_j + \beta d_j]$. Applying an analogon of Smith's rule (Smith, 1956), we minimize this cost component by scheduling the jobs in $\mathcal{E}$ in the interval $[T-p(\mathcal{E}), T]$ in order of non-increasing processing times; the correctness of this rule is easily verified by an interchange argument. The other subproblem is solved by Smith's rule: simply schedule the jobs in $\mathcal{R}$ in non-decreasing order of their processing times in the interval $[0, p(\mathcal{R})]$. In the example, $\mathcal{E} = \emptyset$, and the lower bound is 21α.

A slight improvement of the lower bound is possible. Let $E_{\max}^*$ be the minimum maximum earliness for the jobs in $\mathcal{R}$ if they are processed in the interval $[0, p(\mathcal{R})]$. We compute $E_{\max}^*$ from the minimum-slack-time sequence, that is, the sequence in which the jobs appear in order of non-decreasing values $d_j - p_j$. Avoiding $E_{\max}^*$ requires at least $E_{\max}^*$ units of machine idle time. The lower bound can therefore be improved by $\alpha E_{\max}^*$. If we have stored the shortest-processing-time sequence and the minimum-slack-time sequence, then we compute this lower bound in $O(n)$ time per node. In the example, we have $E_{\max}^* = 4$; hence, the lower bound is 25α. This lower bound approach can only be applied in conjunction with Theorem 2 if $\mathcal{E} = \emptyset$.

Since all jobs in $\mathcal{R}$ are scheduled in the interval $[0, p(\mathcal{R})]$, and since only one early job in $\mathcal{R}$ is taken into account, this lower bound is only effective if the due dates are small relative to the sum of the processing times.

4.2. Second method: relax the machine capacity

Recall that we write the objective function alternatively as $f(\sigma)=(\beta-\alpha)\Sigma_{j=1}^{n} E_j+\alpha\Sigma_{j=1}^{n} T_j+\alpha\Sigma_{j=1}^{n} d_j$ for each $\sigma\in\Omega$. Since the job earlinesses and tardinesses are non-negative by definition, we have that $f(\sigma)\geq\alpha\Sigma_{j=1}^{n} d_j$ for each $\sigma\in\Omega$.

We gain more insight if we derive this bound in the following way. Suppose that the machine can process an infinite number of jobs at the same time; this is a relaxation of the limited capacity of the machine. As $\alpha<\beta$, the optimal schedule has $C_j=d_j$ for each J_j; this gives rise to the lower bound $\alpha\Sigma_{j=1}^{n} d_j$. If no jobs overlap in their execution, then this schedule is feasible and hence optimal for the original problem. For the example, this relaxation gives the lower bound 35α. The corresponding schedule is not feasible: J_2 and J_3 overlap in their execution (see Figure 2).

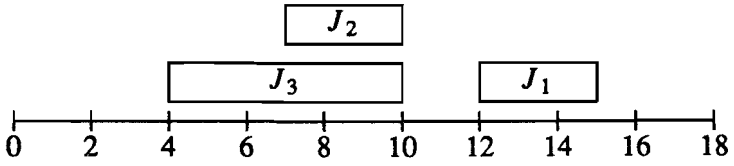

Figure 2. Gantt chart for machine with infinite capacity.

This conflict can be settled by executing J_3 before J_2, or, conversely, J_2 before J_3. If we intend to schedule J_2 after J_3, then we have basically two options: we retain either the completion time of J_3 or the completion time of J_2. For the first option, the additional cost is 3α; for the second option, the additional cost is $3(\beta-\alpha)$. Executing J_2 after J_3 costs therefore at least 3γ extra, where $\gamma=\min\{\alpha, \beta-\alpha\}$. Similarly, we find that executing J_3 after J_2 costs 6γ extra. Hence, the *minimum* additional cost required to settle the overlap is $\min\{3\gamma, 6\gamma\}=3\gamma$. Accordingly, an improved lower bound is 38α.

We now describe a general procedure to improve the lower bound $\alpha\Sigma_{j=1}^{n} d_j$ by taking the overlap between jobs into consideration. Overlap of J_j and J_k ($J_j\neq J_k$) occurs if the intervals $[d_j-p_j, d_j]$ and $[d_k-p_k, d_k]$ overlap. Let $c_{jk}=\gamma\max\{0, d_j-(d_k-p_k)\}$ denote the *additional* cost to execute J_j immediately before J_k; let $\sigma(i)=j$ denote that J_j occupies the ith position in the sequence σ. For any optimal schedule σ, we have that $f(\sigma)\geq\alpha\Sigma_{j=1}^{n} d_j+\Sigma_{j=1}^{n-1} c_{\sigma(j)\sigma(j+1)}$; the last term is the length of the Hamiltonian path $\sigma(1)\cdots\sigma(n)$. The following procedure shows that the Hamiltonian path problem is solvable in $O(n\log n)$ time.

Partition the set of jobs into a set of clusters $Q_1,\ldots,Q_m$ as described above. Let HP_l be the shortest Hamiltonian path for Q_l, and let $c(HP_l)$ denote its length. We have $c(HP_l)=\gamma(p(Q_l)-\max_{J_j, J_k\in Q_l, J_k\neq J_j} c_{jk})$, for each l ($l=1,\ldots,m$). We have also $\Sigma_{j=1}^{n-1} c_{\pi(j)\pi(j+1)}\geq\Sigma_{l=1}^{m} c(HP_l)$ for any sequence π, as can be easily verified. The individual Hamiltonian paths can be combined into one Hamiltonian path of length no more than the sum of the lengths of the separate paths.

4.3. Third method: relax the due dates

4.3.1. The common due date problem

Suppose the due dates have been replaced by a due date d common to all jobs. Consider the following *common due date problem*: for a given d, determine a schedule that minimizes

$$(\beta-\alpha)\sum_{j=1}^{n} E_j+\alpha\sum_{j=1}^{n} T_j+\alpha nd-\beta\sum_{j=1}^{n}\max\{0,d-d_j\}. \qquad \text{(CD)}$$

For any d, the optimal solution value is a lower bound for the original problem, since

$$f(\sigma)=\alpha\sum_{j=1}^{n} C_j+\beta\sum_{j=1}^{n}\max\{0,d_j-C_j\}$$

$$\geq\alpha\sum_{j=1}^{n} C_j+\beta\sum_{j=1}^{n}\max\{0,d-C_j\}-\beta\sum_{j=1}^{n}\max\{0,d-d_j\}$$

$$=(\beta-\alpha)\sum_{j=1}^{n} E_j+\alpha\sum_{j=1}^{n} T_j+\alpha nd-\beta\sum_{j=1}^{n}\max\{0,d-d_j\}.$$

There are two issues involved: (i) how to solve problem (CD)?, and (ii) how to find the value d maximizing the lower bound?

Problem (CD) consists of two parts. The first part is the problem of minimizing $(\beta-\alpha)\Sigma_{j=1}^{n} E_j+\alpha\Sigma_{j=1}^{n} T_j$. If the machine is only available from time 0 onwards and if d is given, then this problem is $\mathcal{NP}$-hard (Hall, Kubiak, and Sethi, 1991; Hoogeveen and Van de Velde, 1991). However, a strong lower bound $L(d)$ is derived by applying Lagrangian relaxation (see Hoogeveen, Oosterhout, and Van de Velde, 1990). The second part is the problem of maximizing the function $G: d\rightarrow \alpha nd-\beta\Sigma_{j=1}^{n}\max\{0,d-d_j\}$; this problem is solvable in polynomial time. Rather than solving problem (CD) to optimality and finding the best d, we maximize the lower bound $L(d)+G(d)$ over d.

First, we derive the best Lagrangian lower bound $L(d)$ for a given d. The derivation proceeds without details; we refer to Hoogeveen, Oosterhout, and van de Velde (1990) for an elaborate treatment. Let $\mathcal{E}$ denote the set of jobs that are not tardy. Since the machine is only available from time 0 onwards, we have the condition that $p(\mathcal{E})\leq d$. We dualize this condition by use of the Lagrangian multiplier $\lambda\geq 0$. For a given $\lambda\geq 0$, the Lagrangian problem is then to find $L(d,\lambda)$, which is the minimum of

$$(\beta-\alpha)\Sigma_{j=1}^{n} E_j+\alpha\Sigma_{j=1}^{n} T_j+\lambda p(\mathcal{E})-\lambda d.$$

The Lagrangian problem is solvable in polynomial time by Emmons's matching algorithm (Emmons, 1987), which proceeds by the concept of positional weights. Straightforward arguments show that there exists an optimal schedule with some job completed exactly on its due date. The weights for the early positions are then λ, $\lambda+(\beta-\alpha)$, $\lambda+2(\beta-\alpha),\ldots,\lambda+(n-1)(\beta-\alpha)$; the smallest weight is for the first position in the schedule. The weights for the tardy positions are $\alpha, 2\alpha,\ldots,n\alpha$; the smallest weight is for the last position in the schedule. Emmons's matching algorithm assigns the job with the jth largest processing time to the position with the jth smallest weight, for $j=1,\ldots,n$. Ties are settled to minimize the amount of work before d. Let σ_λ be the optimal schedule for the Lagrangian problem, and let $W(\sigma_\lambda)$ be the amount of work before d in σ_λ.

The best Lagrangian lower bound $L(d)$ is found as

$$L(d)=\max\{L(d,\lambda) \mid \lambda \geq 0\}.$$

Due to the integrality of α and β, the optimization over $\lambda \geq 0$ may be reduced to the optimization over $\lambda \in \mathbb{N}_0$. The optimal choice for λ can be shown to be such that $W(\sigma_{\lambda-1}) > d \geq W(\sigma_\lambda)$; this choice gives us the Lagrangian lower bound $L(d)$.

We are now able to characterize the function $L: d \rightarrow L(d)$. The function L is continuous and piecewise linear; the value $L(d)$ depends on d only through the choice for λ. Hence, there are at most $\min\{n^2, n\alpha\}$ breakpoints: they correspond to the values $d=W(\sigma_\lambda)$, for $\lambda=0,1,\ldots,n\alpha$. The derivative of the trade-off curve between two consecutive breakpoints, the first corresponding to $W(\sigma_\lambda)$, is equal to $-\lambda$.

The function $G: d \rightarrow \alpha nd - \beta\Sigma_{j=1}^{n} \max\{0, d-d_j\}$ is also continuous and piecewise linear; the breakpoints correspond to the values $d=d_j$, for $j=1,\ldots,n$. The lower bound $L(d)+G(d)$ is therefore also continuous and piecewise linear in d; the value d maximizing this lower bound is found at a breakpoint.

For any given d, $L(d)$ is determined in $O(n\log n)$ time. The function L has $O(\min\{n^2, n\alpha\})$ breakpoints; the corresponding values are computed in $O(n^2)$ time. (Every new breakpoint is derived from the previous one by interchanging some jobs, which requires only constant time, and only $O(n^2)$ interchanges are needed to find all breakpoints.) The function G has $O(n)$ breakpoints. Hence, maximizing $L(d)+G(d)$ over d is achieved in $O(n^2)$ time.

In our 3-job example, we have $d=10$. For the positions after d, the weights are 1,2, and 3; for the positions before d, the weights are 0, 3, and 6. An optimal schedule is depicted in Figure 3. Its objective value is 39α; this happens to be the optimal solution value for the original problem.

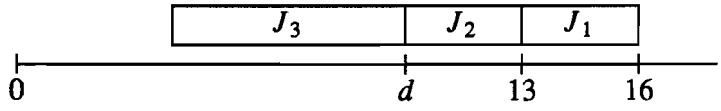

Figure 3. Optimal schedule for the common due date problem.

In a node of the search tree, there are two ways to implement this lower bound procedure. Let $\pi=\pi_1\pi_2$ be the partial schedule associated with the node. Disregarding π, we get the lower bound $f(\pi)+c(\mathcal{J}-\pi)$, where $c(\mathcal{J}-\pi)$ denotes the optimal solution value for the common due date problem for the jobs in $\mathcal{J}-\pi$. However, if π_1 and the optimal schedule for the common due date problem overlap in their execution, then it makes sense to take π_1 into regard. We do this in the following way. First of all, we require that d is common to each $J_j\pi_2$. Subsequently, we solve the common due date problem under the condition that the jobs in π_1 retain their positions. Given the set of positions, it is easy to construct an optimal schedule: assign the jobs in π_1 to the last $|\pi_1|$ positions, and assign the other jobs to the remaining positions according to Emmons's algorithm. Lemma 1 states that we may use the same set of positions as for the case $\pi_1=\emptyset$.

Lemma 1. *The optimal schedule for the common due date problem with the last* $|\pi_1|$ *jobs fixed occupies the* $\bar{n}$ *positions with least positional weights, where* $\bar{n}=n-|\pi_2|$.

Proof. Suppose to the contrary that the optimal schedule σ for the jobs $J_j\pi_2$ does not occupy the $\bar{n}$ positions with least positional weights. Let n_1 jobs in σ be early or just-in-time and let $n_2=\bar{n}-n_1$ jobs in σ be tardy. Suppose the set of optimal weights corresponds to $\bar{n}_1$ positions before d, and to $\bar{n}_2=\bar{n}-\bar{n}_1$ positions after d. Suppose $n_1<\bar{n}_1$. We then transfer the job occupying the n_2th tardy position in σ (the first tardy job) to the (n_1+1)th early position. The latter position is in the optimal set; the former is not. Hence, this transfer reduces the objective value, thereby contradicting the optimality of σ. If $n_1>\bar{n}_1$, then a similar argument applies. □

The common due date lower bound can only be used in conjunction with Theorem 2 if the lower bound is independent from the partial sequence $j\pi$. It is effective if the due dates are close to each other.

4.3.2. The common slack time problem

Consider the special case of the $1||\alpha\Sigma C_j+\beta\Sigma E_j$ problem where all jobs have equal slack time s; i.e., $d_j-p_j=s$ for each J_j $(j=1,\ldots,n)$. This problem has the same features as the common due date problem. The best Lagrangian lower bound is also computed in $O(n\min\{\alpha,n\})$ time; there are the same options to implement the lower bound. The common slack time lower bound is effective if all slack times are close to each other.

4.4. Fourth method: relax the processing times

Again, we consider a special case of the $1||\alpha\Sigma C_j+\beta\Sigma E_j$ problem. Assume that all processing times are equal. Theorem 3 indicates that the *earliest-due-date* sequence (i.e., the sequence with the jobs in order of non-decreasing due dates) is optimal. This special case is solved in $O(n^2)$ time, which is needed to compute the optimal schedule for a given sequence.

Let us return to our original problem. Define $p_{\min}=\min_{1\le j\le n}p_j$. The optimal solution value of the relaxed problem $1|p_j=p_{\min}|\alpha\Sigma C_j+\beta\Sigma E_j$ provides a lower bound for the original problem: each set of job completion times that is feasible for the original problem is also feasible for the relaxed problem and has equal cost.

Given a partial schedule π, let σ be the earliest-due-date sequence for the jobs in $\mathcal{J}-\pi$, and let $g(\sigma)$ be the optimal solution value for the relaxed problem. Disregarding π, we get the lower bound $f(\pi)+g(\sigma)$. We can marginally improve on this lower bound. Suppose we have reindexed the jobs in order of non-decreasing due dates. Corollary 4 indicates that J_n is also scheduled last if we put its processing time equal to $\min\{p_n, p_{\min}+d_n-d_{n-1}\}$. An improved lower bound is therefore given by $f(\pi)+g(\sigma)+\alpha[\min\{p_n, p_{\min}+d_n-d_{n-1}\}-p_{\min}]$.

If the execution of jobs in σ overlap with the execution of jobs in π, then it pays to take π into account. The lower bound is then equal to the cost for the sequence $\sigma\pi$ with the jobs in π still having their original processing times.

Both bounds are computed in $O(n^2)$ time and dominate the lower bound $\alpha\Sigma_{j=1}^{n}d_j$. Only the first version can be used in conjunction with Theorem 2. The common processing time lower bounds are only effective if the processing times are close to each other.

In our 3-job example, we have $p_{\min}=3$, $d_1=15$, and $d_2=d_3=10$. An optimal schedule for the common processing time problem is depicted in Figure 4. Its objective value is 39α; this is equal to the optimal solution value for the original problem.

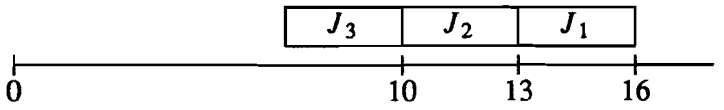

Figure 4. Optimal schedule for the common processing time problem.

4.5. Fifth method: Lagrangian relaxation

The problem of minimizing total inventory cost, referred to as problem (P), can be formulated as follows. Determine values C_j and E_j $(j=1,\ldots,n)$ that minimize

$$\alpha \sum_{j=1}^{n} C_j + \beta \sum_{j=1}^{n} E_j \tag{P}$$

subject to

$$E_j \geq 0, \qquad \text{for } j=1,\ldots,n, \tag{5}$$

$$E_j \geq d_j - C_j, \qquad \text{for } j=1,\ldots,n, \tag{6}$$

$$C_j \geq C_k + p_j \text{ or } C_k \geq C_j + p_k, \qquad \text{for } j,k=1,\ldots,n, j \neq k, \tag{7}$$

$$C_j - p_j \geq 0, \qquad \text{for } j=1,\ldots,n. \tag{8}$$

The conditions (5) and (6) reflect the definition of job earliness, while the conditions (7) ensure that the machine executes at most one job at a time. The conditions (8) express that the machine is available only from time 0 onwards.

We introduce a non-negative vector $\lambda=(\lambda_1,\ldots,\lambda_n)$ of Lagrangian multipliers in order to dualize the conditions (5). For a given vector $\lambda \geq 0$, the Lagrangian problem is to determine the value $L(\lambda)$, which is the minimum of

$$\alpha \sum_{j=1}^{n} C_j + \sum_{j=1}^{n} (\beta - \lambda_j) E_j$$

subject to the conditions (6), (7), and (8). We know that for any given $\lambda \geq 0$ the value $L(\lambda)$ provides a lower bound to problem (P). If $\beta - \lambda_j < 0$ for some J_j, we get $E_j = \infty$, which disqualifies the lower bound. We therefore assume that

$$\lambda_j \leq \beta, \qquad \text{for } j=1,\ldots,n. \tag{9}$$

This, in turn, implies that, for any solution to the Lagrangian problem, conditions (6) hold with equality: $E_j = d_j - C_j$ for each j $(j=1,\ldots,n)$. Hence, the Lagrangian problem, referred to as problem (L_λ), transforms into the problem of minimizing

$$\sum_{j=1}^{n} (\alpha - \beta + \lambda_j) C_j + \sum_{j=1}^{n} (\beta - \lambda_j) d_j \tag{L_λ}$$

subject to

$$C_j \geq C_k + p_j \text{ or } C_k \geq C_j + p_k, \quad \text{for } j, k = 1, \ldots, n, j \neq k, \tag{7}$$

$$C_j - p_j \geq 0, \quad \text{for } j = 1, \ldots, n. \tag{8}$$

If $\alpha-\beta+\lambda_j < 0$ for some J_j, we get $C_j = \infty$, which makes the lower bound rather weak. However, as demonstrated at the beginning of Section 4, we can determine an upper bound T on the job completion times, which implies that

$$C_j \leq T, \quad \text{for } j = 1, \ldots, n. \tag{10}$$

Although the conditions (10) are redundant for the primal problem (P), they are essential to admit values $\lambda_j < \beta-\alpha$. For solving problem (L_λ) under these additional conditions, we first determine the sets of jobs $\mathcal{J}^+ = \{J_j \mid \lambda_j > \beta-\alpha\}$, $\mathcal{J}^- = \{J_j \mid \lambda_j < \beta-\alpha\}$, and $\mathcal{J}^0 = \{J_j \mid \lambda_j = \beta-\alpha\}$. The following theorem stipulates that problem (L_λ) is solved by a simple extension of Smith's rule (Smith, 1956) for solving the $1 \mid\mid \Sigma w_j C_j$ problem; the proof proceeds by an elementary interchange argument.

Theorem 9. *Problem* (L_λ) *with the additional conditions* (10) *is solved by scheduling the jobs in* $\mathcal{J}^+$ *in non-increasing order of ratios* $(\alpha-\beta+\lambda_j)/p_j$ *in the interval* $[0, p(\mathcal{J}^+)]$, *and scheduling the jobs in* $\mathcal{J}^-$ *in non-increasing order of ratios* $(\alpha-\beta+\lambda_j)/p_j$ *in the interval* $[T-p(\mathcal{J}^-), T]$. *The remaining jobs can be scheduled in any order in the interval* $[p(\mathcal{J}^+), T-p(\mathcal{J}^-)]$. □

We are interested in determining the vector $\lambda^* = (\lambda_1^*, \ldots, \lambda_n^*)$ of Lagrangian multipliers that induces the best Lagrangian lower bound. The vector λ^* stems from solving the *Lagrangian dual problem*, referred to as problem (D): maximize

$$L(\lambda) \tag{D}$$

subject to

$$0 \leq \lambda_j \leq \beta, \quad \text{for } j = 1, \ldots, n.$$

Problem (D) is solvable to optimality in polynomial time by use of the ellipsoid method; see Van de Velde (1991). Since the ellipsoid method is very slow in practice, we take our resort to an approximation algorithm for problem (D).

First, we identify the *primitive directional derivatives*. In the solution to the Lagrangian problem (L_λ), the position of J_j depends on the ratio $(\alpha-\beta+\lambda_j)/p_j$; we call this ratio the *relative weight* of J_j. The larger this relative weight, the smaller the completion time of J_j. If other jobs have precisely the same relative weight as J_j, then the exact position of J_j is determined by settling ties. Let now $C_j^+(\lambda)$ denote the earliest possible completion time of J_j in an optimal schedule for problem (L_λ); let $C_j^-(\lambda)$ denote the latest possible completion time of J_j in an optimal schedule for problem (L_λ). If we increase λ_j by $\varepsilon > 0$, then we can choose ε small enough to make sure that at least one optimal schedule for problem (L_λ) remains optimal; for a proof, see Van de Velde (1991). In fact, all such optimal schedules must have J_j completed on time $C_j^+(\lambda)$. If we increase λ_j by such a sufficiently small $\varepsilon > 0$, then the Lagrangian objective value is affected by $\varepsilon(C_j^+(\lambda) - d_j)$. The primitive directional derivative for increasing λ_j, as denoted by $l_j^+(\lambda)$, is therefore simply

$$l_j^+(\lambda) = C_j^+(\lambda) - d_j, \quad \text{for } j = 1, \ldots, n.$$

Hence, if $l_j^+(\lambda) > 0$, then increasing λ_j is an ascent direction: we get an improved lower bound by moving some scalar step size along this direction. In a similar fashion, we derive that the primitive directional derivative for decreasing λ_j, denoted by $l_j^-(\lambda)$, is

$$l_j^-(\lambda) = d_j - C_j^-(\lambda), \text{ for } j = 1, \ldots, n.$$

If $l_j^-(\lambda) > 0$, then decreasing λ_j is an ascent direction. Note that directional derivatives may not exist at the boundaries of the feasible region of λ; for instance, $l_i^-(\lambda)$ is undefined for $\lambda = (0, \ldots, 0)$, for any $i = 1, \ldots, m$.

Second, we determine an appropriate step size $\Delta > 0$ to move by along a chosen ascent direction. We compute the step size that takes us to the first point where the corresponding primitive directional derivative is no longer positive. If no such point exists, then we choose the step size as large as possible while maintaining feasibility.

Suppose $l_j^+(\lambda) > 0$: J_j is tardy in any optimal schedule for problem (L_λ). Increasing λ_j, thereby putting J_j earlier in the schedule, is an ascent direction. We distinguish the cases $p_j - d_j > 0$, $p_j - d_j = 0$, and $p_j - d_j < 0$. Consider the case $p_j - d_j > 0$. Hence, J_j is unavoidably tardy, and $l_j^+(\lambda) > 0$ for all $\lambda \geq 0$ with $\lambda_j < \beta$. Therefore, we take the step size $\Delta = \beta - \lambda_j$. Accordingly, we must also have that $\lambda_j^* = \beta$; otherwise, increasing λ_j^* would be an ascent direction. If $p_j = d_j$, then there exists an optimal solution to problem (D) with $\lambda_j^* = \beta$. Find $\mathcal{T} = \{J_j \mid p_j \geq d_j\}$. We have proven the following result.

Theorem 10. *There exists an optimal solution for the Lagrangian dual problem* (D) *with* $\lambda_j^* = \beta$ *for each* $J_j \in \mathcal{T}$. □

Suppose now $p_j < d_j$. The step size Δ must satisfy $\lambda_j + \Delta \leq \beta$. We identify the first job in the schedule, say, J_k, for which $C_k - p_k + p_j \leq d_j$. Since $p_j < d_j$, such a J_k always exists. If J_j is scheduled in J_k's position, then J_j is not tardy. Hence, if there were no upper bound on λ, then increasing λ_j would be an ascent direction up to the point where the relative weight of J_j becomes equal to the relative weight of J_k. Hence, the maximum step size along this ascent direction is the largest value Δ such that

$$(\alpha - \beta + \lambda_j + \Delta)/p_j \leq (\alpha - \beta + \lambda_k)/p_k, \text{ and}$$

$$\lambda_j + \Delta \leq \beta.$$

Let now $\bar{\lambda} = (\lambda_1, \ldots, \lambda_j + \Delta, \ldots, \lambda_n)$. Suppose $\bar{\lambda}_j + \Delta < \beta_j$. Since the relative weights for all jobs but J_j have remained the same, optimal solutions for the problems $(L_{\bar{\lambda}})$ and (L_λ) exist with the same jobs scheduled before J_k. Now J_j and J_k have equal relative weights: in any optimal solution to problem $(L_{\bar{\lambda}})$, J_j can be scheduled before J_k or after J_k. If J_j is scheduled before J_k, then J_j is not tardy; if J_j is scheduled after J_k, then J_j is not early. Hence, we have that $C_j^+(\bar{\lambda}) \leq d_j \leq C_j^-(\bar{\lambda})$; the step size Δ has taken us to the first point where the primitive directional derivative for increasing λ_j is no longer positive. If $\bar{\lambda}_j = \beta$, then the step size has been chosen as large as possible.

Suppose now $l_j^-(\lambda) < 0$: J_j is early in any optimal schedule for problem (L_λ). Decreasing λ_j, thereby deferring J_j, is an ascent direction. We distinguish the cases $d_j > T$, $d_j = T$, and $d_j < T$. Consider the case $d_j > T$; hence, J_j is unavoidably early, and $l_j^-(\lambda) > 0$ for all λ with $\lambda_j > 0$. Therefore, we choose the step size as large as possible: $\Delta = \lambda_j$. Accordingly, we also must have that $\lambda_j^* = 0$; otherwise, decreasing λ_j^* would be an ascent direction. If $d_j = T$, then there exists an optimal schedule to problem (D) with $\lambda_j^* = 0$. Identify $\mathcal{E} = \{J_j \mid d_j \geq T\}$. We have proven the following result.

Theorem 11. *There exists an optimal solution for the Lagrangian dual problem* (D) *with* $\lambda_j^* = 0$ *for each* $J_j \in \mathcal{E}$. □

Consider now the case $d_j < T$. The procedure to compute the appropriate step size Δ proceeds in a similar fashion as above. We identify some J_k as the first job in the schedule with $C_k \geq d_j$. If J_j is scheduled in J_k's position, then J_j is not early. Hence, if there were no lower bound on λ, then decreasing λ_j would be an ascent direction up to the point where the relative weight of J_j becomes equal to the relative weight of J_k. Hence, the maximum step size along this ascent direction is the largest value Δ for which

$$(\alpha-\beta+\lambda_j-\Delta)/p_j \geq (\alpha-\beta+\lambda_k)/p_k, \text{ and}$$

$$\lambda_j-\Delta \geq 0.$$

Let $\bar{\lambda}=(\lambda_1,\ldots,\lambda_j-\Delta,\ldots,\lambda_n)$. Suppose $\bar{\lambda}_j > 0$. Since the relative weights for all jobs but J_j have remained the same, optimal solutions for the problems $(L_{\bar{\lambda}})$ and (L_λ) exist with the same jobs scheduled after J_k. Since J_j and J_k have now equal weights, J_j can be scheduled after J_k or before J_k in any optimal schedule for problem $(L_{\bar{\lambda}})$. If J_j is scheduled after J_k, then J_j is not early; if J_j is scheduled before J_k, then J_j is not tardy. Hence, we find that $C_j^+(\bar{\lambda}) \leq d_j \leq C_j^-(\bar{\lambda})$. If $\bar{\lambda}_j = 0$, then the step was taken as large as possible.

Termination of the ascent direction procedure occurs at some $\bar{\lambda}$ where all existing primitive directional derivatives are non-positive. If all primitive directional derivatives exist at such a $\bar{\lambda}$, we have

$$C_j^+(\bar{\lambda}) \leq d_j \leq C_j^-(\bar{\lambda}), \quad \text{for } j=1,\ldots,n.$$

These termination conditions also apply to λ^*, since they are necessary for optimality. They are, however, not sufficient for optimality; hence, termination may occur having $\bar{\lambda} \neq \lambda^*$, i.e., before finding the optimal vector of Lagrangian multipliers. Before implementing the ascent direction algorithm, we make use of this fact to decompose the Lagrangian dual problem (D) into two subproblems. This decomposition is achieved by partitioning $\mathcal{J}$ into four subsets, including the sets $\mathcal{T}$ and $\mathcal{E}$ we already identified.

Consider some job $J_j \in \mathcal{J}-\mathcal{E}$ with $d_j > p(\mathcal{J}-\mathcal{E})$. If $\lambda_j > \beta-\alpha$, then J_j will be early in any optimal solution to problem (L_λ). This means that $l_j^-(\lambda) > 0$, and hence we must have that $0 \leq \lambda_j^* \leq \beta-\alpha$. The set $\mathcal{F}$ of jobs that share this property is determined by the following procedure.

Partitioning Algorithm 1

Step 0. $\mathcal{F} \leftarrow \emptyset$, and reindex the jobs in $\mathcal{J}-\mathcal{E}$ according to non-increasing due dates. Let $k \leftarrow 1$.

Step 1. If $k > n-|\mathcal{E}|$ or if $d_k < p(\mathcal{J}-\mathcal{E}-\mathcal{F})$, then stop. Else $\mathcal{F} \leftarrow \mathcal{F} \cup \{J_k\}$.

Step 2. Set $k \leftarrow k+1$; go to Step 1.

Suppose some job $J_j \in \mathcal{F}$ exists with $d_j > T-p(\mathcal{E})$. If we let $\lambda_j = \beta-\alpha$, then $C_j^-(\lambda) < d_j$; hence, decreasing λ_j is an ascent direction. Decreasing λ_j gives $(\alpha-\beta+\lambda_j)/p_j < 0$, as a result of which the execution of J_j interferes with the execution of the jobs in $\mathcal{E}$. We now partition the set $\mathcal{F}$ into subsets $\mathcal{F}_1$ and $\mathcal{F}_2$ $(\mathcal{F}=\mathcal{F}_1 \cup \mathcal{F}_2)$ such that $d_j \leq T-p(\mathcal{E} \cup \mathcal{F}_2)$ for each $J_j \in \mathcal{F}_1$, and such that $d_j > T-p(\mathcal{E} \cup \mathcal{F}_2)$ for each $J_j \in \mathcal{F}_2$. To achieve this, we use the following partitioning procedure; it is similar to the first one.

Partitioning Algorithm 2

Step 0. Put $\mathcal{F}_2 \leftarrow \emptyset$, let $P \leftarrow T-p(\mathcal{E})$, and reindex the jobs in $\mathcal{F}$ according to non-increasing due dates. Let $k \leftarrow 1$.

Step 1. If $k > |\mathcal{F}|$, then stop. If $d_k \le P$, then let $\mathcal{F}_1 \leftarrow \{J_k, \ldots, J_{|\mathcal{F}|}\}$, and stop. Otherwise, $\mathcal{F}_2 \leftarrow \mathcal{F}_2 \cup \{J_k\}$, and set $P \leftarrow P - p_k$.
Step 2. Set $k \leftarrow k+1$; go to Step 1.

Let $\mathcal{R} = \mathcal{J} - \mathcal{T} - \mathcal{E} - \mathcal{F}$.

Theorem 12. *For each $J_j \in \mathcal{F}_1$, we have that $\lambda_j^* = \beta - \alpha$.*

Proof. Since we have $p(\mathcal{T} \cup \mathcal{R}) \le d_j \le T - p(\mathcal{E} \cup \mathcal{F}_2)$, the result follows. □

At this stage, we can decompose the Lagrangian dual problem (D) into two subproblems. Since $(\alpha - \beta + \lambda_j^*)/p_j = 0$ for each $J_j \in \mathcal{F}_1$, the jobs in $\mathcal{F}_1$ do not interfere with the execution of the other jobs. However, $\mathcal{T}$ and $\mathcal{R}$ interfere with each other, and $\mathcal{E}$ and $\mathcal{F}_2$ interfere with each other. On the one hand, we have the dual problem restricted to the sets $\mathcal{T}$ and $\mathcal{R}$; on the other hand, we have the dual problem restricted to the sets $\mathcal{F}_2$ and $\mathcal{E}$. In each optimal schedule for problem (D), the jobs in $\mathcal{T}$ and $\mathcal{R}$ are scheduled in the interval $[0, p(\mathcal{T} \cup \mathcal{R})]$, and the jobs in $\mathcal{F}$ and $\mathcal{E}$ are scheduled in the interval $[T - p(\mathcal{E} \cup \mathcal{F}_2), T]$. We give step-wise descriptions of the ascent direction algorithms for these two subproblems. Both are based upon the primitive directional derivatives and the step sizes we discussed earlier. The jobs in $\mathcal{F}_1$ are scheduled somewhere in the interval $[p(\mathcal{T} \cup \mathcal{R}), T - p(\mathcal{E} \cup \mathcal{F}_2)]$; they are left out of consideration. We introduce some new notation. Let $(L_\lambda^{\mathcal{R} \cup \mathcal{T}})$ and $(L_\lambda^{\mathcal{E} \cup \mathcal{F}_2})$ denote the Lagrangian problem restricted to the set $\mathcal{R} \cup \mathcal{T}$ and to the set $\mathcal{E} \cup \mathcal{F}_2$; let $L^{\mathcal{R} \cup \mathcal{T}}(\lambda)$ and $L^{\mathcal{E} \cup \mathcal{F}_2}(\lambda)$ denote their optimal solution values.

Ascent Direction Algorithm for the Set $\mathcal{T} \cup \mathcal{R}$
Step 0. For each $J_j \in \mathcal{T}$, set $\lambda_j \leftarrow \lambda_j^* = \beta$; for each $J_j \in \mathcal{R}$, set $\lambda_j \leftarrow \beta$. Solve $(L_\lambda^{\mathcal{R} \cup \mathcal{T}})$, settling ties arbitrarily; compute the job completion times.
Step 1. For each $J_j \in \mathcal{R}$, do the following:
(a) If $C_j^-(\lambda) < d_j$, identify J_k as the first job in the schedule with $C_k \ge d_j$. Compute the largest value Δ such that

$$(\alpha - \beta + \lambda_j - \Delta)/p_j \ge (\alpha - \beta + \lambda_k)/p_k, \text{ and} \tag{11}$$

$$\lambda_j - \Delta \ge \beta - \alpha. \tag{12}$$

Decrease λ_j by Δ, reposition J_j according to its new relative weight, and update the job completion times.
(b) If $C_j^+(\lambda) > d_j$, identify J_k that is the first job in the schedule with $C_k - p_k + p_j \le d_j$. Compute the largest value for Δ such that

$$(\alpha - \beta + \lambda_j + \Delta)/p_j = (\alpha - \beta + \lambda_k)/p_k, \text{ and}$$

$$\lambda_j + \Delta \le \beta.$$

Increase λ_j by Δ, reposition J_j according to its new relative weight, and update the job completion times.
Step 2. If no multiplier adjustment has taken place, then compute $L^{\mathcal{R} \cup \mathcal{T}}(\lambda)$ and stop. Otherwise, go to Step 1.

Theorem 13. *The procedure described above generates a series of monotonically increasing values* $L^{\mathcal{R}\cup\mathcal{T}}(\lambda)$.

Proof. First, consider some $J_j \in \mathcal{R}$ with $C_j^-(\lambda) < d_j$: decreasing λ_j is an ascent direction. For brevity, we let $\mu_j = \alpha - \beta + \lambda_j$ for each j ($j = 1, \ldots, |\mathcal{R}\cup\mathcal{T}|$). We reindex the jobs in order of non-increasing values μ_j/p_j, settling all ties arbitrarily except for J_j: we give J_j the largest index possible. Accordingly, we obtain the sequence $(J_1, \ldots, J_{|\mathcal{R}\cup\mathcal{T}|})$, which is optimal for problem $(L_\lambda^{\mathcal{R}\cup\mathcal{T}})$, with job completion times $C_1, \ldots, C_{|\mathcal{R}\cup\mathcal{T}|}$. We note that $C_j = C_j^-(\lambda)$. Let Δ be the step size computed as prescribed in the ascent direction algorithm, and let $\bar{\lambda} = (\lambda_1, \ldots, \lambda_j - \Delta_1, \ldots, \lambda_{|\mathcal{R}\cup\mathcal{T}|})$.

We distinguish the case that condition (11) holds with equality from the case that condition (12) holds with equality. Consider the first case; accordingly let J_k be the job specified in the ascent direction procedure. In more detail, the sequence under consideration is $(J_1, \ldots, J_{j-1}, J_j, J_{j+1}, \ldots, J_{k-1}, J_k, J_{k+1}, \ldots, J_{|\mathcal{R}\cup\mathcal{T}|})$; an optimal sequence for problem $(L_{\bar{\lambda}}^{\mathcal{R}\cup\mathcal{T}})$ is then $(J_1, \ldots, J_{j-1}, J_{j+1}, \ldots, J_k, J_j, J_{k+1}, \ldots, J_{|\mathcal{R}\cup\mathcal{T}|})$. The job completion times for the latter sequence can conveniently be expressed in terms of $C_1, \ldots, C_{|\mathcal{R}\cup\mathcal{T}|}$. We now prove that $L^{\mathcal{R}\cup\mathcal{T}}(\bar{\lambda}) > L^{\mathcal{R}\cup\mathcal{T}}(\lambda)$. We have

$$L^{\mathcal{R}\cup\mathcal{T}}(\bar{\lambda}) = \sum_{i=1}^{j-1} \mu_i C_i + (\mu_j - \Delta)(C_j^-(\lambda) + \sum_{i=j+1}^{k} p_i) + \sum_{i=j+1}^{k} \mu_i (C_i - p_j) + \sum_{i=k+1}^{|\mathcal{R}\cup\mathcal{T}|} \mu_i C_i + \sum_{i=1}^{|\mathcal{R}\cup\mathcal{T}|} (\beta - \lambda_i) d_i + \Delta d_j$$

$$= L^{\mathcal{R}\cup\mathcal{T}}(\lambda) - p_j \sum_{i=j+1}^{k} \mu_i + \mu_j \sum_{i=j+1}^{k} p_i - \Delta(C_j^-(\lambda) + \sum_{i=j+1}^{k} p_i - d_j)$$

$$= L(\lambda) - p_j \sum_{i=j+1}^{k-1} \mu_i + \mu_j \sum_{i=j+1}^{k-1} p_i - \Delta(C_j^-(\lambda) + \sum_{i=j+1}^{k-1} p_i - d_j) + (\mu_j - \Delta) p_k - p_j \mu_k.$$

Note that $(\mu_j - \Delta)/p_j = \mu_k/p_k$; hence, we have $(\mu_j - \Delta)p_k - p_j\mu_k = 0$. This implies that

$$L(\bar{\lambda}) \geq L(\lambda) + p_j \sum_{i=j+1}^{k-1} \left[p_i(\mu_j/p_j - \mu_i/p_i) \right] - \Delta(C_j^-(\lambda) + \sum_{i=j+1}^{k-1} p_i - d_j).$$

Since $d_j > C_j^-(\lambda) + \Sigma_{i=j+1}^{k-1} p_i$, $\mu_j/p_j > \mu_i/p_i$ for each i ($i = j+1, \ldots, k-1$), and $\Delta > 0$, we have that $L^{\mathcal{R}\cup\mathcal{T}}(\bar{\lambda}) > L^{\mathcal{R}\cup\mathcal{T}}(\lambda)$.

Now assume that the condition (12) holds with equality and the condition (11) does not: $\Delta = \alpha - \beta + \lambda_j$. This implies that J_j will now be placed after some job J_h, with $j \leq h < k$. For this case, the second sequence is $(J_1, \ldots, J_{j-1}, J_{j+1}, \ldots, J_h, J_j, J_{h+1}, \ldots, J_k, \ldots, J_{|\mathcal{R}\cup\mathcal{T}|})$. We perform a similar analysis as above to obtain

$$L^{\mathcal{R}\cup\mathcal{T}}(\bar{\lambda}) = L^{\mathcal{R}\cup\mathcal{T}}(\lambda) - p_j \sum_{i=j+1}^{h} \mu_i + \mu_j \sum_{i=j+1}^{h} p_i - \Delta(C_j^-(\lambda) + \sum_{i=j+1}^{h} p_i - d_j)$$

$$= L^{\mathcal{R}\cup\mathcal{T}}(\lambda) + p_j \sum_{i=j+1}^{h} \left[p_i(\mu_j/p_j - \mu_i/p_i) \right] - \Delta(C_j^-(\lambda) + \sum_{i=j+1}^{h} p_i - d_j).$$

At this point, similar arguments as before apply to show that $L^{\mathcal{R}\cup\mathcal{T}}(\bar{\lambda}) > L^{\mathcal{R}\cup\mathcal{T}}(\lambda)$.

Second, consider the case that $C_j^+(\lambda) > d_j$ for some $J_j \in \mathcal{R}$; increasing λ_j is an ascent direction. Let Δ be the desired step size, computed as described in the ascent direction algorithm. The proof to show that $L^{\mathcal{R}\cup\mathcal{T}}(\lambda_1, \ldots, \lambda_j + \Delta, \ldots, \lambda_{|\mathcal{R}\cup\mathcal{T}|}) > L^{\mathcal{R}\cup\mathcal{T}}(\lambda_1, \ldots, \lambda_j, \ldots, \lambda_{|\mathcal{R}\cup\mathcal{T}|})$ follows the same lines as above. □

Ascent Direction Algorithm for the Set $\mathcal{F}_2 \cup \mathcal{E}$

Step 0. Set $\lambda_j \leftarrow \beta - \alpha$ for each $J_j \in \mathcal{F}_2$, and $\lambda_j \leftarrow \lambda_j^* = 0$ for each $J_j \in \mathcal{E}$. Solve $(L_\lambda^{\mathcal{E}\cup\mathcal{F}_2})$, settling ties arbitrarily; compute the job completion times.

Step 1. For each $J_j \in \mathcal{F}_2$, do the following:

(a) If $C_j^-(\lambda) < d_j$, identify J_k as the first job in the schedule with $C_k \geq d_j$. Compute the largest value Δ such that

$$(\alpha - \beta + \lambda_j - \Delta)/p_j \geq (\alpha - \beta + \lambda_k)/p_k, \text{ and}$$

$$\Delta \leq \lambda_j.$$

Decrease λ_j by Δ, reposition J_j according to its new relative weight, and update the job completion times.

(b) If $C_j^+(\lambda) > d_j$, identify J_k that is the first job in the schedule with $C_k \leq d_j + p_k - p_j$. Compute the largest value for Δ such that

$$(\alpha - \beta + \lambda_j + \Delta)/p_j = (\alpha - \beta + \lambda_k)/p_k, \text{ and}$$

$$\lambda_j + \Delta \leq \beta - \alpha.$$

Increase λ_j by Δ, reposition J_j according to its new relative weight, and update the job completion times.

Step 2. If no multiplier adjustment has taken place, then compute $L^{\mathcal{R}\cup\mathcal{T}}(\bar{\lambda})$ and stop. Otherwise, go to Step 1.

Theorem 14. *The procedure described above generates a series of monotonically increasing values* $L^{\mathcal{R}\cup\mathcal{T}}(\bar{\lambda})$.

Proof. The proof proceeds along the same lines as the proof of Theorem 13. □

For each $J_j \in \mathcal{J} - \mathcal{F}_1$, let C_j and $\bar{\lambda}_j$ denote the completion time and the Lagrangian multiplier upon termination of the appropriate ascent direction algorithm. We note that $\bar{\lambda}_j = \beta_j$ for each $J_j \in \mathcal{T}$, $\bar{\lambda}_j = \beta - \alpha$ for each $J_j \in \mathcal{F}_1$, and $\bar{\lambda}_j = 0$ for each $J_j \in \mathcal{E}$. Hence, the overall Lagrangian lower bound is given by

$$L(\bar{\lambda}) = \sum_{J_j \in \mathcal{T}} \alpha C_j + \sum_{J_j \in \mathcal{F}_1} \alpha d_j + \sum_{J_j \in \mathcal{E}} \left[(\alpha - \beta) C_j + \beta d_j \right] + \sum_{J_j \in \mathcal{R}\cup\mathcal{F}_2} \left[(\alpha - \beta + \bar{\lambda}) C_j - (\beta - \bar{\lambda}_j) d_j \right]$$

5. Computational results

The algorithm was coded in the computer language C; the experiments were conducted on a Compaq-386/20 Personal Computer. The algorithm was tested on instances with 8, 10, 12, 15, and 25 jobs. The processing times were generated from the uniform distribution [10,100]. The due dates were generated from the uniform distribution $[P(1-T-R/2), P(1-T+R/2)]$, where $P=\Sigma_{j=1}^{n} p_j$ and where R and T are parameters. For both parameters, we considered the values 0.2, 0.4, 0.6, 0.8, and 1.0. This procedure to generate due dates parallels the procedure described by Potts and Van Wassenhove (1985) for the weighted tardiness problem. For each combination of T, P, and n, we generated 5 instances. Each instance was considered with $\alpha=1$ and with β running from 2 to 5.

The general impression was that instances become difficult with smaller values of T, with smaller values of R, and with smaller values of β. A small value of T induces relative large due dates, implying that the machine will be idle for some time before processing the first job. A small value of R induces due dates that are close to each other; it is then harder to partition the jobs. A large value of β implies that earliness is severely penalized; most jobs will therefore be tardy. Accordingly, the instances with $T=0.2, R=0.2$, and $\beta=5$ are the hardest; the instances with $T=1.0, R=1.0$, and $\beta=2$ are the easiest.

Table 2 exhibits a summary of our computational results; we only report the results for the instances with T and R equal. It shows that instances with up to 10 jobs are easy. For $n=12$, the instances with $T=R=0.2$ require already considerable effort. For $n=20$, only the choice $T=R=1.0$ induces instances that are solvable within reasonable time limits. It is likely, however, that the performance of the algorithm is considerably enhanced by fine-tuning the algorithm to specific instances. Currently, all lower bounds are computed in each node of the tree; Lagrangian relaxation, for instance, is useless for instances with $T=R=0.2$.

6. Conclusions

Although machine idle time is a practical instrument to reduce inventory cost, a considerable lack of theoretical analysis of related machine scheduling problems exists. Within this context, we have addressed the $1||\alpha\Sigma C_j+\beta\Sigma E_j$ problem for the case that $\alpha<\beta$. It is a very difficult problem from a practical point of view.

Acknowledgement

The authors like to thank Jan Karel Lenstra for his helpful comments.

References

A.V. Aho, J.E. Hopcroft, and J.D. Ullman (1982). *Data Structures and Algorithms*, Addison-Wesley, Reading, Massachusetts.

K. Baker and G. Scudder (1990). Sequencing with earliness and tardiness penalties: a review. *Operations Research 38*, 22-36.

J. Du and J. Y-T. Leung (1990). Minimizing total tardiness on one machine is *NP*-hard. *Mathematics of Operations Research 15*, 483-495.

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

T.D. Fry and G. Keong Leong (1987a). Single machine scheduling: a comparison of two solution procedures. *Omega 15*, 277-282.

n	T,R	β=2 nodes	β=2 sec	β=3 nodes	β=3 sec	β=4 nodes	β=4 sec	β=5 nodes	β=5 sec
8	0.2	417	2	406	2	301	2	58	1
8	0.4	131	1	198	1	185	1	31	1
8	0.6	34	1	48	1	29	1	5	1
8	0.8	23	1	37	1	14	1	8	1
8	1.0	20	1	36	1	33	1	15	1
10	0.2	2438	8	2525	9	2088	7	484	2
10	0.4	266	2	689	3	570	3	202	2
10	0.6	123	1	110	1	88	1	52	1
10	0.8	126	1	122	1	107	1	64	1
10	1.0	109	1	140	1	78	1	40	1
12	0.2	30182	103	26676	106	18358	78	10487	48
12	0.4	15176	66	20756	100	15613	75	10391	50
12	0.6	212	2	262	2	53	1	10	1
12	0.8	380	2	576	4	300	2	170	1
12	1.0	432	2	527	3	226	2	96	1
15	0.2	-	-	-	-	-	-	(2)	-
15	0.4	(3)	-	(2)	-	(2)	-	(30	-
15	0.6	1414	10	2407	17	927	7	339	2
15	0.8	1665	13	1865	15	1647	14	540	5
15	1.0	493	6	402	17	2063	17	1082	9
20	0.2	-	-	-	-	-	-	-	-
20	0.4	-	-	-	-	-	-	-	-
20	0.6	7991	80	13169	136	5529	62	2048	24
20	0.8	8183	85	7244	84	4016	55	1318	21
20	1.0	5127	49	5243	41	2191	32	651	12

Table 2. Computational results. For each combination of n (n=8,10,12,15,20), of T and R (T=R=0.2,0.4,0.6,0.8,1.0), and of β (β=2,3,4,5), we present the average number of nodes and the average number of seconds; the average was computed over 5 instances. All averages were rounded up to the nearest integer. The sign '-' indicates that not all instances of this particular combination could be solved without examining more than 100,000 nodes.

T.D. Fry and G. Keong Leong (1987b). A bi-criterion approach to minimizing inventory costs on a single machine when early shipments are forbidden. *Computers and Operations Research 14*, 363-368.

M.R. Garey, R.E. Tarjan, and G.T. Wilfong (1988). One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research 13*, 330-348.

R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics 5*,

287-326.

N.G. Hall, W. Kubiak, and S.P. Sethi (1991). Earliness-tardiness scheduling problems, II: Deviation of completion times about a restrictive common due date. *Operations Research 39*, 847-856.

J.A. Hoogeveen (1990). *Minimizing maximum earliness and maximum lateness on a single machine*, Report BS-R9001, CWI, Amsterdam.

J.A. Hoogeveen, H. Oosterhout, and S.L. van de Velde (1992). New lower and upper bounds for scheduling around a small common due date. To appear in *Operations Research*.

J.A. Hoogeveen and S.L. van de Velde (1992). Polynomial-time algorithms for single-machine multicriteria scheduling. To appear in *Operations Research*.

J.A. Hoogeveen and S.L. van de Velde (1991). Scheduling around a small common due date. *European Journal of Operational Research 55*, 237-242.

L.G. Khachiyan (1979). A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR 244*, 1093-1096 (English translation: *Soviet Mathematics Doklady 20*, 191-194).

E.L. Lawler and J.M. Moore (1969). A functional equation and its application to resource allocation and sequencing problems. *Management Science 16*, 77-84.

J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics 1*, 343-362.

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

C.H. Papadimitriou and K. Steiglitz (1982). *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, New Jersey.

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

W.E. Smith (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly 1*, 59-66.

S.L. van de Velde (1991). *Machine scheduling and Lagrangian relaxation*, Doctoral Thesis, CWI, Amsterdam.

List of COSOR-memoranda - 1992

Number	Month	Author	Title
92-01	January	F.W. Steutel	On the addition of log-convex functions and sequences
92-02	January	P. v.d. Laan	Selection constants for Uniform populations
92-03	February	E.E.M. v. Berkum H.N. Linssen D.A. Overdijk	Data reduction in statistical inference
92-04	February	H.J.C. Huijberts H. Nijmeijer	Strong dynamic input-output decoupling: from linearity to nonlinearity
92-05	March	S.J.L. v. Eijndhoven J.M. Soethoudt	Introduction to a behavioral approach of continuous-time systems
92-06	April	P.J. Zwietering E.H.L. Aarts J. Wessels	The minimal number of layers of a perceptron that sorts
92-07	April	F.P.A. Coolen	Maximum Imprecision Related to Intervals of Measures and Bayesian Inference with Conjugate Imprecise Prior Densities
92-08	May	I.J.B.F. Adan J. Wessels W.H.M. Zijm	A Note on "The effect of varying routing probability in two parallel queues with dynamic routing under a threshold-type scheduling"
92-09	May	I.J.B.F. Adan G.J.J.A.N. v. Houtum J. v.d. Wal	Upper and lower bounds for the waiting time in the symmetric shortest queue system
92-10	May	P. v.d. Laan	Subset Selection: Robustness and Imprecise Selection
92-11	May	R.J.M. Vaessens E.H.L. Aarts J.K. Lenstra	A Local Search Template (Extended Abstract)
92-12	May	F.P.A. Coolen	Elicitation of Expert Knowledge and Assessment of Imprecise Prior Densities for Lifetime Distributions
92-13	May	M.A. Peters A.A. Stoorvogel	Mixed H_2/H_∞ Control in a Stochastic Framework

Number	Month	Author	Title
92-14	June	P.J. Zwietering E.H.L. Aarts J. Wessels	The construction of minimal multi-layered perceptrons: a case study for sorting
92-15	June	P. van der Laan	Experiments: Design, Parametric and Nonparametric Analysis, and Selection
92-16	June	J.J.A.M. Brands F.W. Steutel R.J.G. Wilms	On the number of maxima in a discrete sample
92-17	June	S.J.L. v. Eijndhoven J.M. Soethoudt	Introduction to a behavioral approach of continuous-time systems part II
92-18	June	J.A. Hoogeveen H. Oosterhout S.L. van der Velde	New lower and upper bounds for scheduling around a small common due date
92-19	June	F.P.A. Coolen	On Bernoulli Experiments with Imprecise Prior Probabilities
92-20	June	J.A. Hoogeveen S.L. van de Velde	Minimizing Total Inventory Cost on a Single Machine in Just-in-Time Manufacturing