PROD. TYPE: COM
PP: 1-12 (col.fig.: Nil)
CAOR1522 DTD VER: 5.0.1
ED: Joji
PAGN: Murthy.N – SCAN: Unknown
ARTICLE IN PRESS
Available online at www.sciencedirect.com
SCIENCE DIRECT
ELSEVIER
Computers & Operations Research ▮▮▮ (▮▮▮) ▮▮▮–▮▮▮
computers & operations research
www.elsevier.com/locate/dsw

Minimizing the earliness–tardiness costs on a single machine

Jakub Bauman[a], Joanna Józefowska[b,*]

[a]Baj-soft, ul. Gwarna 11 m. 4 61-207 Poznań, Poland
[b]Institute of Computing Science, Poznań University of Technology, ul. Piotrowo 3a, 60-965 Poznań, Poland

Abstract

In this paper the one-machine scheduling problem with linear earliness and tardiness costs is considered. The cost functions are job dependent and asymmetric. The problem consists of two sub-problems. The first one is to find a sequence of jobs and the second one is to find the job completion times that are optimal for the given sequence. We consider the second sub-problem and propose an algorithm solving the problem in $O(n \log n)$ time.

Keywords: Scheduling; Single machine; Earliness–tardiness costs; Just-in-time

1. Introduction

In modern enterprises the control of the production process encompasses the whole supply chain. One of the benefits of such approach is the reduction of inventory costs. The supplier is supposed to deliver goods as close to the required date as possible. This concept is often called Just-in-Time (JIT) production. The JIT concept for manufacturing has induced a new type of machine scheduling problem in which both early and tardy completions of jobs are penalised. The earliness costs include inventory costs emerging if a product is completed before its due date. The tardiness costs relate to penalty costs emerging if production is completed after the due date. Both earliness and tardiness costs are assumed to be functions of the relevant distance of job's completion time from its due date. Linear and non-linear functions are considered. The objective is to minimise the total cost. The emerging objective function is a non-regular performance measure as defined by Conway et al. [1]. It means that the penalty function does not necessarily increase with the increase of job completion times. In consequence, it may be appropriate

* Corresponding author. Tel.: +48 61 8790790; fax: +48 61 8771525.
E-mail addresses: jb@baj-soft.com (J. Bauman), Joanna.Jozefowska@cs.put.poznan.pl (J. Józefowska).

0305-0548/$ - see front matter © 2005 Published by Elsevier Ltd.
doi:10.1016/j.cor.2005.02.037

to insert idle time between jobs. In general, the problem is NP-hard even for one machine as shown by Garey et al. [2]. The problem was further considered by Yano and Kim [3], Abdul-Razaq and Potts [4], Szwarc [5], Ow and Morton [6,7], Fry et al. [8]. A review of this and similar problems was given by Baker and Scudder [9].

In this paper we consider the earliness–tardiness problem with linear job dependent and asymmetric cost functions on a single machine.

This problem can be decomposed into two sub-problems: to find a sequence of jobs and to find optimal completion times of jobs in the given sequence (i.e. to find a schedule). In general, an optimal schedule for a given sequence of jobs can be found by solving a linear programming problem. However, more efficient procedures can be developed. Namely, Garey et al. [2] proposed a simple procedure called GTW for a special case of the problem with job independent and symmetric earliness and tardiness costs. The procedure can be implemented to run in $O(n \log n)$ time. In this paper we propose an algorithm of the same complexity to solve the problem with asymmetric and job-dependent costs.

In Section 2 the problem is formulated and some properties of optimal solutions are shown. A concept of a cost-increase function ΔK, basic for the optimization procedure is introduced in Section 3. Section 4 contains the description of the scheduling algorithm, the proof of the correctness of the algorithm and its worst case complexity. Finally in Section 5 some conclusions and directions for further research are outlined.

2. Problem formulation

Let us consider n non-preemptable jobs to be scheduled on a single machine, each job i having a due date d_i, and processing time p_i, $i = 1, \ldots, n$. Without loss of generality we can assume that the processing times and the due dates are integers. The machine can handle no more than one job at a time and it is continuously available from time zero onwards only. Assume that there is a feasible schedule S (i.e. such that the jobs do not overlap in their execution and no job starts its processing before time zero) in which C_i is the completion time of job i, $i = 1, \ldots, n$. We assume that the earliness, as well as tardiness, costs are linear functions of the deviation of job's completion time C_i from its due date d_i. The earliness cost is positive only if $d_i - C_i > 0$, otherwise it is zero. On the other hand, the tardiness cost is positive only if $C_i - d_i > 0$, otherwise it is zero. In general, the total earliness and tardiness cost of schedule S may be calculated as follows:

$$f(S) = \sum_{i=1}^{n} (\alpha_i \max\{0, d_i - C_i\} + \beta_i \max\{0, C_i - d_i\}), \tag{1}$$

where α_i is the cost of job i being completed one time unit before its due date, and β_i is the cost of job i being completed one time unit after its due date $i = 1, \ldots, n$. Our goal is, obviously, to find a feasible schedule S^* with the minimum value of function $f(S)$.

As we have mentioned in the introduction this NP-hard problem can be decomposed into two sub-problems: to build a sequence of jobs and to find completion times of jobs in the sequence. The second problem can be solved efficiently using linear programming or an algorithm proposed by Chrétienne and Sourd [10]. An optimal sequence can be chosen using an exhaustive search over the set of all permutations of jobs. Such approach is obviously computationally ineffective since the number of sequences to be considered equals $n!$.

Let us assume that the sequence of jobs is given. The problem is to find an optimal vector of completion times of jobs. Observe that unlike for regular schedule performance measures (like $C_{\max}$ or $L_{\max}$) inserting machine idle time may be desirable. In general, the vector of optimal completion times may be found by solving the following LP problem:

Minimize

$$f = \sum_{i=1}^{n} (\alpha_i C_i^{+} + \beta_i C_i^{-}) \tag{2}$$

Subject to:

$$C_i \geqslant C_{i-1} + p_i, \quad i = 2, \ldots, n, \tag{3}$$

$$C_1 \geqslant p_1, \tag{4}$$

$$C_i^{+} \geqslant d_i - C_i, \quad i = 1, \ldots, n, \tag{5}$$

$$C_i^{-} \geqslant C_i - d_i, \quad i = 1, \ldots, n, \tag{6}$$

$$C_i^{-} \geqslant 0, \quad i = 1, \ldots, n, \tag{7}$$

$$C_i^{+} \geqslant 0, \quad i = 1, \ldots, n. \tag{8}$$

In practice, solving problem (2)–(8) may be time consuming. Chrétienne and Sourd [10] proposed a general procedure to find optimal schedules in case of convex cost functions and precedence constraints between jobs. As a special case, the above mentioned problem can be solved in $O(n \log n)$ time. Below we will propose a more explicit algorithm for the same problem which runs also in $O(n \log n)$ time.

3. Properties of the cost function

Let us assume that the sequence of jobs is fixed. Our goal is to find a vector of completion times of jobs, such that $C_1 \leqslant C_2 \leqslant \cdots \leqslant C_n$ and the value of the cost function (1) is minimal for this sequence (i.e. permutation of jobs). It is natural to schedule the jobs iteratively, adding one job at a time to the schedule. Let us denote by C_i^k the completion time of job i in a feasible schedule of the jobs $1, 2, \ldots, k$, $k \leqslant n$. Let σ_{k-1}^{*} be an optimal schedule, i.e. the vector of completion times $[C_1^{(k-1)*}, C_2^{(k-1)*}, \ldots, C_{k-1}^{(k-1)*}]$ of the sequence consisting of the first $k-1$ jobs, $2 \leqslant k < n$ and $K(\sigma_{k-1}^{*})$ be the cost of schedule σ_{k-1}^{*}. While scheduling job k we have to consider the following two cases:

(i) $C_{k-1}^{(k-1)*} + p_k \leqslant d_k$,
(ii) $C_{k-1}^{(k-1)*} + p_k > d_k$.

In the first case, obviously, $\sigma_k^* = [C_1^{(k-1)*}, C_2^{(k-1)*}, \ldots, C_{k-1}^{(k-1)*}, d_k]$ is an optimal schedule for the sequence of jobs $1, 2, \ldots, k$. Thus the cost $K(\sigma_k^*) = K(\sigma_{k-1}^*)$ and

$$C_1^{k*} = C_1^{(k-1)*}, C_2^{k*} = C_2^{(k-1)*}, \ldots, C_{k-1}^{k*} = C_{k-1}^{(k-1)*}, C_k^{k*} = d_k.$$

In the second case, however, either σ_{k-1}^* is left unchanged and job k is scheduled late at time $C_k^k = C_{k-1}^{(k-1)*} + p_k > d_k$ or we have to find a schedule of jobs $1, 2, \ldots, k$, completed at $C_{k-1}^{k-1} < C_{k-1}^{(k-1)*}$ and to schedule job k so that it completes at $\max\{C_{k-1}^{k-1} + p_k, d_k\}$.

Below we will show the idea of finding a schedule of jobs $1, 2, \ldots, k$, $k \leqslant n$ shorter than the optimal one (recall that an optimal schedule minimizes the earliness–tardiness cost which is not a regular measure). Let $\sigma_k = [C_1^k, C_2^k, \ldots, C_k^k]$, be a schedule of k jobs such that $\sum_{i=1}^k p_i \leqslant C_k^k < C_k^{k*}$, and constructed from the schedule σ_k^* as follows. We start from the last job k. Its new completion time is $C_k^k < C_k^{k*}$. Now, if $C_k^k - p_k \geqslant C_{k-1}^{k*}$, job $k-1$, as well as its predecessors have the same completion times in σ_k^* and in σ_k, i.e. $C_i^k = C_i^{k*}$, $i = 1, 2, \ldots, k-1$. If, however $C_k^k - p_k < C_{k-1}^{k*}$, then $C_{k-1}^k = C_k^k - p_k$. We continue this way as long as $C_i^k - p_i < C_{i-1}^{k*}$ or $i = 2$. We have assumed that $\sum_{i=1}^k p_i \leqslant C_k^k$, so $C_1^k = C_2^k - p_2 \geqslant p_1$ and the obtained schedule is feasible. More formally this algorithm is described below as the LEFT_SHIFT procedure.

Procedure LEFT_SHIFT

begin for $i := k-1$ **step** -1 **to** 1 **do** $C_i^k := \min\{C_i^{k*}, C_{i+1}^k - p_{i+1}\}$ **end.**

Observe the following property of the LEFT_SHIFT procedure.

Property 1. Let σ be a schedule of jobs $1, 2, \ldots, k$, $k \leqslant n$ of length C. Consider $C_2 < C_1 < C$. Let us use the LEFT_SHIFT procedure to obtain from schedule σ, a schedule σ_1 of length C_1. Now, let us apply the LEFT_SHIFT procedure to σ_1 in order to obtain a schedule σ_2 of length C_2. Finally, let us apply LEFT_SHIFT procedure to schedule σ in order to obtain a schedule σ_3 of length C_2. It is easy to see that $\sigma_2 = \sigma_3$.

Since $K(\sigma_{k-1}^*)$ is the cost of an optimal schedule of jobs $1, 2, \ldots, k$, $k \leqslant n$, we have $K(\sigma_{k-1}^*) \leqslant K(\sigma_{k-1})$. However, if $C_{k-1}^{(k-1)*} \geqslant C_{k-1}^{k-1} \geqslant d_k - p_k$, then $\beta_k(C_{k-1}^{k-1} + p_k - d_k) \leqslant \beta_k(C_{k-1}^{(k-1)*} + p_k - d_k)$. Thus it may be favorable to shorten the schedule σ_{k-1}^*. Observe that in general, $\sigma_{k-1} = [C_1^{k-1} - x_1^{k-1}, C_2^{k-1} - x_2^{k-1}, \ldots, C_{k-1}^{k-1} - x_{k-1}^{k-1}]$, where $0 \leqslant x_i^{k-1} \leqslant C_i^{(k-1)*} - \sum_{j=1}^i p_j$ for $i = 1, 2, \ldots, k-1$. Concluding, if

$$\begin{aligned} &K(C_1^{(k-1)*} - x_1^{(k-1)}, C_2^{(k-1)*} - x_2^{(k-1)}, \ldots, C_{k-1}^{(k-1)*} - x_{k-1}^{k-1}) + \beta_k(C_{k-1}^{k-1} + p_k - d_k) \\ &\quad \leqslant K(C_1^{(k-1)*}, C_2^{(k-1)*}, \ldots, C_{k-1}^{(k-1)*}) + \beta_k(C_{k-1}^{(k-1)*} + p_k - d_k), \end{aligned}$$

it is favorable to shorten the schedule σ_{k-1}^*.

Let us now consider the function $\Delta K_k(x_1^k, x_2^k, \ldots, x_k^k)$, defined as the difference between the cost of schedule σ_k, of length $C_k^k = C_k^{k*} - x_k^k$ obtained from σ_k^* using the LEFT_SHIFT procedure and an optimal schedule σ_k^*.

$$\Delta K_k(x_1^k, x_2^k, \ldots, x_k^k) = K(C_1^{k*} - x_1^k, C_2^{k*} - x_2^k, \ldots, C_k^{k*} - x_k^k) - K(C_1^{k*}, C_2^{k*}, \ldots, C_k^{k*}).$$

Since σ_k is obtained according to the LEFT_SHIFT procedure, the values $x_i^k, i = 1, 2, \ldots, k$, are related. Namely, $x_i^k = \max\{0, x_{i+1}^k - (C_{i+1}^{k*} - C_i^{k*} - p_{i+1})\},\ i = 1, 2, \ldots, k-1$, and $x_k^k = C_k^{k*} - C_k^k$. Thus, the value of function ΔK_k depends only on the value x_k^k and we have:

$$\Delta K_k(x_k^k) = K(C_1^{k*} - x_1^k, C_2^{k*} - x_2^k, \ldots, C_k^{k*} - x_k^k) - K(C_1^{k*}, C_2^{k*}, \ldots, C_k^{k*}),$$

where $x_i^k = \max\{0, x_{i+1}^k - (C_{i+1}^{k*} - C_i^{k*} - p_{i+1})\},\ i = 1, 2, \ldots, k-1$, and $x_k^k = C_k^{k*} - C_k^k$. Concluding, it is enough to find the value x_k^k minimizing:

$$\Delta K_k(x_k^k) + \beta_{k+1}(C_k^{k*} + p_{k+1} - x_k^k - d_{k+1}). \tag{9}$$

Further on we always consider the shortest schedule of all the schedules of same cost, i.e. the greatest value of x_k^k minimizing (9).

Now, it remains to calculate x_k^k. Let us start with constructing the function $\Delta K_k(x_k^k)$. If the first job is late (i.e. $p_1 > d_1$), $\Delta K_1(x_1^1)$ grows to infinity for any $x_1^1 > 0$. Otherwise, clearly, the function $\Delta K_1(x_1^1) = \alpha_1 x_1^1$, where $0 \leqslant x_1^1 \leqslant d_1 - p_1$, and $C_1^{1*} = d_1$. Observe that function $\Delta K_k(x_k^k)$ can be constructed from $\Delta K_{k-1}(x_{k-1}^{k-1})$ in the following way. Let $y_k = d_k - p_k - C_{k-1}^{(k-1)*}$. If $y_k \geqslant 0$ then

$$\Delta K_k(x_k^k) = \begin{cases} \alpha_k x_k^k & \text{if } 0 \leqslant x_k^k \leqslant y_k, \\ \Delta K_{k-1}(x_k^k - y_k) + \alpha_k x_k^k & \text{if } y_k \leqslant x_k^k \leqslant d_k - \sum_{i=1}^k p_i, \end{cases}$$

else (if $y_k < 0$)

$$\Delta K_k(x_k^k) = \begin{cases} \Delta K_{k-1}(x_k^k) - \beta_k x_k^k & \text{if } 0 \leqslant x_k^k \leqslant -y_k, \\ \Delta K_{k-1}(x_k^k) - \beta_k(-y_k) + \alpha_k(y_k + x_k^k) & \text{if } -y_k \leqslant x_k^k \leqslant d_k - \sum_{i=1}^k p_i. \end{cases}$$

Observe, that in the latter case, the function $\Delta K_k(x_k^k)$ may attain its minimum at some point $x_k^{k*} > 0$. Then, of course, we do not need to consider values $x < x_k^{k*}$ any further, because adding consecutive jobs we can only be interested in decreasing the length of the current schedule. A graph of function $\Delta K_k(x_k^k)$ is presented in Fig. 1. The dotted line has to be deleted before scheduling the next job.

Lemma 1. *Function $\Delta K_k(x_k^k)$ is piecewise linear, convex, and increasing $k = 1, \ldots, n$.*

The proof follows easily from the construction of function $\Delta K_k(x_k^k)$.

Lemma 2 shows that given a sequence of k jobs, a schedule of length $C_k^k < C_k^{k*}$ where, $\sum_{i=1}^k p_i \leqslant C_k^k$, (obviously, any schedule of length $C_k^k < \sum_{i=1}^k p_i$ is infeasible) obtained from o_k^* according to the LEFT_SHIFT procedure is a schedule with minimal cost of all the schedules of this length.

Lemma 2. *If o_k^* is an optimal schedule for a given sequence of k jobs and C_k^{k*} is the completion time of the last job in o_k^* then schedule σ_k such that the completion time of the last job in σ_k is $C_k^k, \sum_{i=1}^k p_i \leqslant C_k^k < C_k^{k*}$, obtained according to the LEFT_SHIFT procedure is a schedule with minimal cost of all the schedules for the given sequence of jobs completed at C_k^k.*

Proof. It follows from the assumption $\sum_{i=1}^k p_i \leqslant C_k^k < C_k^{k*}$ that a schedule of length C_k^k exists. Let us now prove that schedule σ_k obtained according to the LEFT_SHIFT procedure is a schedule with minimal

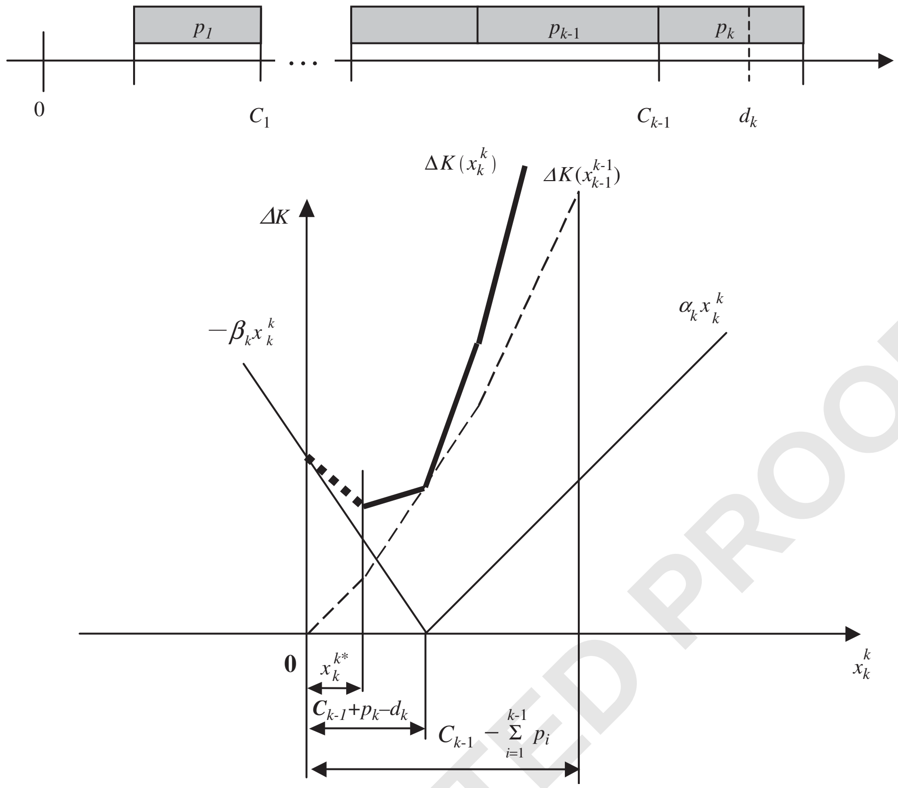

Fig. 1. Function $\Delta K(x_k^k)$ (the case of $d_k < C_{k-1}^{k-1} + p_k$ and large β_k).

cost of all the schedules for the given sequence of jobs of length C_k^k. The Lemma holds, obviously, for a single job.Let us assume that it is true for a sequence of jobs $1, 2, \ldots, k, k \leqslant n$. We will show by a contradiction that it holds for a sequence of k jobs $1, 2, \ldots, k, k \leqslant n$. Let us assume hat there exists a schedule σ_k' of length C_k^k, such that $K(\sigma_k') < K(\sigma_k)$. Since $C_k'^k = C_k^k$, the cost of scheduling the last job is identical in both the schedules. Let us consider two cases:

(i) $C_k^{k*} - p_k > C_k^k - p_k \geqslant C_{k-1}^{k*}$,
(ii) $C_k^k - p_k < C_{k-1}^{k*}$.

Observe that in the first case, according to the LEFT_SHIFT procedure, $C_i^k = C_i^{(k-1)*}$, for $1 \leqslant i \leqslant k-1$, so the schedule of $k-1$ jobs is optimal. Moreover, since there is an idle time scheduled before job j_k in σ_k^*, we know that $C_k^{k*} = d_k$ and consequently job j_k is early in σ_k' and σ_k. Thus,

$$K(\sigma_k') < K(\sigma_k) = K(\sigma_{k-1}^*) + \alpha_k(C_k^{k*} - C_k^k). \tag{10}$$

Since the cost of scheduling job j_k is the same in σ_k' and σ_k, it follows from (10) that there exists a schedule of $k-1$ jobs of cost less than $K(\sigma_{k-1}^*)$. This is a contradiction.

Let us now consider the second case.

Since the schedule $\sigma_k^* = [C_1^{k*}, C_2^{k*}, \ldots, C_k^{k*}]$ is an optimal schedule of k jobs, then the schedule $\sigma_{k-1} = [C_1^{k*}, C_2^{k*}, \ldots, C_{k-1}^{k*}]$, of the first $k-1$ jobs in σ_k^* is a schedule with minimal cost of all the schedules for the given sequence of jobs of length $C_k^{k*} - p_k$. Now let us apply the LEFT_SHIFT procedure to the schedule σ_{k-1} to obtain a schedule of length $C_k^k - p_k$. Due to the Property 1 and our inductive assumption, the resulting schedule σ_k^S is a schedule with minimal cost of all the schedules for the given sequence of $k-1$ jobs of length $C_k^k - p_k$. Since the cost of scheduling job k is the same in σ_k' and σ_k, it follows from (10) that there exists a schedule of $k-1$ jobs of length $C_k^k - p_k$ with cost less than the cost of the schedule σ_k^S, what is a contradiction. □

Further, we will call the points at which the slope of function $\Delta K_k(x_k^k)$ changes—the characteristic points.

Lemma 3. *The maximum number of characteristic points* (*i.e. points in which the function* $\Delta K_n(x_n^n)$ *changes its slope*) *is equal to* $n+1$.

Proof. Observe that the change of the slope of function $\Delta K_n(x_n^n)$ takes place only at points at which a job changes its status from being late to being early. Each job can change its status only once and there are n jobs in the final schedule. The $(n+1)$st point is at $x_n^n = C_n^{n*} - \sum_{i=1}^n p_i$, where the coefficient goes to infinity since no job can be started before time zero. Thus there are at most n intervals with different slope of function $\Delta K_n(x_n^n)$.

Lemma 4. *The slope of function* $\Delta K_n(x_n^n)$ *in an interval* $I_k = (H_k, H_{k+1})$ *between two consecutive characteristic points* $H_k, H_{k+1},\ k = 0, 1, \ldots, p < n$, *can be calculated as* $\sum_{i \in E_{I_k}} \alpha_i - \sum_{i \in L_{I_k}} \beta_i$ *where* E_{I_k} *is the set of jobs being early and* L_{I_k} *is the set of jobs being late if* $x \in I_k$.

This lemma follows directly from the construction of function $\Delta K_n(x_n^n)$. Observe that by the definition of the characteristic points in any interval each job is either early or late.

4. Algorithm

We assume that the order of jobs is given. The algorithm adds one job to the schedule at each iteration finding optimal completion times of jobs already scheduled. Thus an optimal solution of a subset of jobs is found at each iteration. Finally an optimal schedule for n jobs is obtained.

It follows from Fig. 1 that it is convenient to extend the interval of feasible values of x at the left-hand side. Thus we will calculate the characteristic points as negative values. At iteration k the sequence of characteristic points $H_l < H_{l-1}, \ldots, H_1 < H_0 = 0, l \leqslant k$ is known as well as the completion time C_{k-1} of job $k-1$, assuming $C_0 = 0$. We create iteratively a vector of coefficients corresponding to the characteristic points $\gamma(H_i),\ i = 0, 1, \ldots, l$ with $\gamma(H_0) = 0$. At iteration k we calculate $H_{\text{new}} = H_l + (C_{k-1} + p_k - d_k)$. Let us consider two cases:

(a) $H_{\text{new}} \leqslant H_l$,
(b) $H_{\text{new}} > H_l$.

Table 1
Values of the variables after the first iteration, $C_1 = 5$

i	H_i	$\gamma(H_i)$
1	−3	2

(a) If $H_{\text{new}} \leqslant H_l$ then $C_k = d_k$ and a new characteristic point $H_{l+1} = H_{\text{new}}$, is added with $\gamma(H_{l+1}) = \sigma_k$. The special case where $H_{\text{new}} = H_l$ can be easily identified and in this case no new point is added but $\gamma'(H_l) = \gamma(H_l) + \sigma_k$. We pass to iteration $k + 1$.
(b) If $H_{\text{new}} > H_l$ then the sequence of characteristic points is updated as follows. Again two cases are distinguished:

(b1) $H_{\text{new}} < 0$,
(b2) $H_{\text{new}} \geqslant 0$.

(b1) If $H_{\text{new}} < 0$, we insert point H_{new} at the right place (preserving the order) in the sequence of characteristic points. If there exists H_j such that $H_{\text{new}} = H_j$ then we do not create a new point but calculate $\gamma'(H_j) := \gamma(H_j) + \gamma(H_{\text{new}})$, otherwise a new characteristic point H_{new}, with $\gamma(H_{\text{new}}) = \alpha_k + \beta_k$ is created and the number of characteristic points is increased by 1.
(b2) If $H_{\text{new}} \geqslant 0$ no new point is added.

After updating the sequence of characteristic points we calculate $\gamma'(H_l) = \gamma(H_l) - \beta_k$. If $\gamma'(H_l) \leqslant 0$, we remove H_l from further consideration ($l := l - 1$) and analyze the characteristic points from H_l through H_1. At each point H_i we calculate $\gamma'(H_i) = \gamma(H_i) + \gamma(H_{i+1})$. If $\gamma'(H_i) \leqslant 0$ then we remove this point from further consideration (decreasing l) and consider the next characteristic point, otherwise we stop with $l \geqslant 0$ being the current number of characteristic points. Finally we calculate $C_k = \sum_{j=1}^{k} p_j - H_l$ and we pass to iteration $k + 1$.

Notice that completion times of jobs are not updated, even if we remove characteristic points (which means shifting the schedule left). Thus, after adding the last job it is necessary to calculate the optimal completion times C_n^*, $k = 1, 2, \ldots, n$. Obviously, C_n is the optimal completion time of job n, i.e. $C_n^* = C_n$. We calculate the remaining optimal completion times according to the following formula: $C_k^* = \min(C_{k+1}^* - p_{k+1}, C_k)$, $k = 1, 2, \ldots, n - 1$.

More formally the algorithm is presented in the Appendix.

Scheduling a single job requires at most $O(\log n)$ steps. This is the case when a job is late and a new characteristic point has to be inserted at the right order in the list of characteristic points. This can be obviously executed in $(\log n)$ time. Thus, the computational complexity of the algorithm is $O(n \log n)$.

Let us consider the following example.

Example 1. $\mathbf{p} = [2,\ 5,\ 4,\ 3]$, $\mathbf{d} = [5,\ 13,\ 15,\ 17]$, $\boldsymbol{\alpha} = [2,\ 1,\ 3,\ 2]$, $\boldsymbol{\beta} = [1,\ 1,\ 2,\ 1]$.

In the first iteration the first job is scheduled. $C_0 + p_1 - d_1 < 0$, so we take $H_1 = 0 + (0 + 2 - 5) = -3$. Remaining parameters are given in Tables 1–4.

For the second job $C_1 + p_2 - d_2 < 0$ and $H_2 = -3 + (5 + 5 - 13) = -6$.

Let us now consider the third job. $C_2 + p_3 - d_3 \geqslant 0$, so $H_3 = -6 + (13 + 4 - 15) = -4$. This characteristic point has to be inserted between H_1 and H_2. First $\gamma(-6) = -2 + 1 = -1$ and $\gamma(-4) = 3 + 2 = 5$. Since

Table 2
Values of the variables after the second iteration, $C_2 = 13$

i	H_i	$\gamma(H_i)$
1	-3	2
2	-6	1

Table 3
Values of the variables after the third iteration, $C_3 = 15$

i	H_i	$\gamma(H_i)$
1	-3	2
2	-4	$5 - 1 = 4$

Table 4
Values of the variables after the fourth iteration, $C_4 = 18$

i	H_i	$\gamma(H_i)$
1	-3	$2 + 1 + 2 = 5$
2	-4	$4 - 1 = 3$

$\gamma(-6) < 0$, we remove H_2 and update $\gamma(-4) = 5 - 1 = 4$. Finally, we set $H_2 = -4$ and the points are ordered appropriately.

Finally, for the fourth job we have $C_3 + p_4 - d_4 = 15 + 3 - 17 = 1 > 0$, so $H_4 = -4 + 15 + 3 - 17 = -3$. The characteristic point H_4 coincides with the point H_2, so only the coefficients of the cost function have to be updated.

Now we have to calculate the completion times of jobs. Job 4 completes at $C_4 = 18$, thus $C_3 = \min\{C_4 - p_4, C_3^*\} = \min\{15, 15\} = 15$; $C_2 = \min\{C_3 - p_3, C_2^*\} = \min\{11, 13\} = 11$ and $C_1 = \min\{C_2 - p_2, C_1^*\} = \min\{6, 5\} = 5$.

We will show now that the algorithm finds an optimal schedule for a given sequence of n jobs. Let us consider a schedule of a given sequence of jobs and let C_i, be the completion time of job i, $i = 1, 2, \ldots, n$.

Theorem 1. *The algorithm finds an optimal schedule for a given sequence of jobs.*

Proof. By induction on k. The schedule is certainly optimal for $k = 1$. Namely, if $H_{\text{new}} < 0$ we find $H_1 = H_{\text{new}}$ and $C_k = d_k$, so the cost of scheduling job 1 is zero. The schedule is optimal. If $H_{\text{new}} \geqslant 0$, we do not create any new characteristic point and $C_1 = p_1$. Although the job is late, no feasible schedule exists where job 1 completes before C_1. Also in this case the schedule obtained by the algorithm is optimal.

Assuming that the theorem is true for any sequence of $k - 1$ jobs we will prove that it holds for any sequence of k jobs.

Let $\sigma_{k-1} = \lfloor C_1^{k-1}, C_2^{k-1}, \ldots, C_k^{k-1} \rfloor$ be a schedule obtained applying the algorithm for a sequence of jobs $1, 2, \ldots, k - 1$, while $\sigma_k = \lfloor C_1^k, C_2^k, \ldots, C_k^k \rfloor$ a schedule obtained applying the algorithm for the sequence $1, 2, \ldots, k-1, k$. From our inductive assumption σ_{k-1} is optimal for the sequence of jobs $1, 2, \ldots, k-1$. If we apply the algorithm to the sequence $1, 2, \ldots, k-1, k$, then iterations 1 through $k-1$ are identical as for the sequence $1, 2, \ldots, k-1$. After iteration $k-1$, we have l characteristic points with corresponding coefficients $\gamma(H_i),\ i = 1, \ldots, l$, and a vector $[C_1, C_2, \ldots, C_{k-1}]$ from which we obtain the optimal completion times for the sequence $1, 2, \ldots, k-1$, where $C_{k-1} = C_{k-1}^{k-1}$. We will consider two cases:

(a) $C_{k-1}^{k-1} + p_k \leqslant d_k$,
(b) $C_{k-1}^{k-1} + p_k > d_k$.

(a) In the first case $C_k = d_k$. Thus the cost of scheduling job k is zero, and $K(\sigma_k) = K(\sigma_{k-1})$. Since the schedule σ_{k-1} is optimal, also σ_k is optimal.

(b) In the second case the minimum of the cost function is found as follows. Observe that at each characteristic point H_j considered for removal we calculate exactly $\gamma(H_j) = \sum_{i \in E_{I_j}} \alpha_i - \sum_{i \in L_{I_j}} \beta_i$, where E_{I_j} is the set of jobs being early and L_{I_j} is the set of jobs being late if we shift the job k by $x \in I_j$, where $I_j = [H_j, H_{j+1})$. Thus according to Lemma 4 it is the slope of the function $\Delta K_k(x_k^k)$ in the interval I_j.

Again two cases have to be considered:

(b1) $\gamma(H_l) \geqslant \beta_k$,
(b2) $\gamma(H_l) < \beta_k$.

(b1) In this case any shift increases the value of $\Delta K_k(x_k^k)$, so the optimal schedule is obtained for $x = 0$. According to the algorithm the last characteristic point remains unchanged, so $C_k = \sum_{j=1}^{k} p_j - H_s = C_{k-1}^{k-1} + p_k$ which in fact corresponds to $x = 0$. Thus the schedule σ_k is optimal for the sequence of jobs $1, 2, \ldots, k$.

(b2) If $\gamma(H_l) < \beta_k$, the function ΔK decreases in interval $[H_l, H_{l-1})$, so it does not attain its minimum at H_l which can be removed from further consideration. However the coefficient $\gamma(H_{l-1})$ at the next characteristic point is updated. In the next steps we consider the consecutive characteristic points, each time calculating $\gamma(H_j) + \gamma(H_{j-1})$. If $\gamma(H_j) + \gamma(H_{j-1}) \leqslant 0$, the coefficient $\gamma(H_{j-2})$ is updated and the characteristic point H_{j-1} is removed from further consideration. Finally, at H_{l+1} at the latest, we reach the first characteristic point at which $j - \gamma(H_s) > 0$. From this point on function ΔK increases, so at H_s function ΔK attains its minimum. We find $C_k^k = C_k = \sum_{i=1}^{k} p_i - H_s$, where $H_s = \min\{H_l\}$. Since it is the minimum of ΔK, the schedule σ_k is optimal for the sequence of jobs $1, 2, \ldots, k$. This completes the proof. □

5. Conclusions

In this paper we have proposed an $O(n \log n)$ algorithm to solve the problem of scheduling a given sequence of nonpreemptive jobs with individual due dates to minimize the total earliness–tardiness cost. The cost functions considered are linear job dependent and asymmetric. The developed algorithm will be used in branch and bound as well as tabu search procedures for finding an optimal sequence of jobs.

Acknowledgements

We are very grateful to an anonymous referee for very constructive remarks on improving the readability of the paper.

This research has been supported by the Polish State Committee for Scientific Research Grant no. KBN 3 T11F 025 28.

Appendix

The following additional notation is assumed in the description of the algorithm:
l—current number of characteristic points;
H_k, $k = 1, \ldots, l$—the value of the kth characteristic point, $H_1 < H_2 < \cdots < H_l$;
γ_k, $k = 1, \ldots, l$—the coefficient assigned to the kth characteristic point;
C_k—completion time of the last job at the kth iteration
P—the sum of completion times at the current iteration.

ALGORITHM
begin {initialize:} $C_0 := 0$; $l := 0$; $H_0 := 0$; $\gamma_0 := 0$; $P := 0$;
  for $k := 1$ **to** n **do**
  begin $x := C_{k-1} + p_k - d_k$; $P := P + p_k$;
    if $x \leqslant 0$ **then** {add a new characteristic point $H_{l+1} < H_l$}
      begin
        if $x < 0$ **then**
        begin $l := l + 1$;
          $H_l := H_{l-1} + x$;
          $\gamma_l := 0$;
        end;
        $\gamma_l := \gamma_l + \alpha_k$;
        $C_k := d_k$;
      end;
      else {job is late}
      begin $H_{\text{new}} := H_{l+x}$;
          if $H_{\text{new}} < 0$ **then**
          begin *Insert* H_{new} *in the appropriate position in the* sequence of characteristic points.
            if (*there is* H_j *such that* $H_{\text{new}} = H_j$) **then**
              do not create a new point but $\gamma_j := \gamma_j + \gamma_{\text{new}}$;
            else $\gamma_{\text{new}} := \alpha_k + \beta_k$; $l := l + 1$;
          end;
          $\gamma_l := \gamma_l - \beta_k$;
          $i := l$;
          while ($\gamma_i \leqslant 0$ **and** $i > 0$) **do**

begin $\gamma_{i-1} := \gamma_{i-1} + \gamma_i$;
$C_k := P - H_{i-1}$;
$i := i - 1$;
$l := l - 1$;
end;
end;
end;
avail $:= C_n$;
for $k := n$ **to** 1 **step** -1 **do**
begin if $C_k \geqslant$ avail **then** $C_k :=$ avail **else** avail $:= C_k$;
avail $:=$ avail $- p_k$;
end;
end;

References

[1] Conway RW, Maxwell WL, Miller LW. Theory of scheduling. Reading, MA: Addison-Wesley; 1967.
[2] Garey M, Tarjan R, Wilfong G. One-processor scheduling with symmetric earliness and tardiness penalties. Mathematics of Operations Research 1988;13:330–48.
[3] Yano C, Kim Y. Algorithms for single machine scheduling problems minimizing tardiness and earliness. Technical report #86-40, Department of Industrial Engineering, University of Michigan, Ann Arbor, USA, 1986.
[4] Abdul-Razaq T, Potts C. Dynamic programming state-space relaxation for single-machine scheduling. Journal of the Operational Research Society 1988;39:141–52.
[5] Szwarc W. Minimizing absolute lateness in single machine scheduling with different due dates. Working Paper, University of Wisconsin, Milwaukee, USA, 1988.
[6] Ow P, Morton T. Filtered beam search in scheduling. International Journal of Production Research 1988;26:35–62.
[7] Ow P, Morton T. The single machine early–tardy problem. Management Science 1989;35:177–91.
[8] Fry T, Darby-Dowman K, Armstrong R. Single machine scheduling to minimize mean absolute lateness. Working Paper, College of Business Administration, University of South Carolina, Columbia, USA, 1988.
[9] Baker K, Scudder G. Sequencing with earliness and tardiness penalties: a review. Operations Research 1990;38:22–36.
[10] Chrétienne P, Sourd F. PERT scheduling with convex cost functions. Theoretical Computer Science 2003;292:145–64.