Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 191 (2008) 577–581
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Short Communication

Single-machine scheduling with waiting-time-dependent due dates

Christos Koulamas *, George J. Kyparisis

Department of Decision Sciences and Information Systems, Florida International University, Miami, FL 33199, United States

Received 14 September 2006; accepted 14 August 2007
Available online 28 August 2007

Abstract

We consider a single-machine scheduling problem in which due dates are linear functions of the job waiting-times and the objective is to minimize the maximum lateness. An optimal sequence is constructed by implementing an index-based priority rule for a fixed value of the due date normalizing constant k. We determine in polynomial time all the k value ranges so that the optimal sequence remains the same within each range. The optimal due dates are computed as linear functions of the global optimal value of k. The overall procedure is illustrated in a numerical example.

Keywords: Scheduling; Single machine; Lateness; Due date assignment

1. Introduction

In many practical situations involving scheduling decisions the job due dates are assigned internally by the scheduler based on certain shop characteristics. Cheng and Gupta (1989) provide a comprehensive survey of scheduling research involving due date assignment decisions. One way of assigning job due dates is to take into account both job characteristics and the current shop status; e.g., the JIS method assigns due dates based on the number of jobs in the system (Weeks, 1979) and the JIQ method assigns due dates based on current queue lengths (Eilon and Chowdhury, 1976). The simulation results of Eilon and Chowdhury (1976) and Ragatz and Mabert (1985) demonstrate the merits of these due date assignment methods.

The objective of this paper is to construct an optimal sequence and assign the optimal due dates analytically in a single-machine setting when due dates are linear functions of the job waiting-times and the objective is to minimize the maximum job lateness. To our knowledge, our results are the first analytical results for this type of problem. Specifically, we assume that in a non-preemptive single-machine setting all jobs from a given batch are available and ready for processing at the same time (e.g., at time zero). Let p_j denote the processing time of job J_j and d_j denote the baseline due date of J_j which is based exclusively on the characteristics of J_j. Then, the actual waiting-time-adjusted due date of J_j can be defined as $d'_j = d_j + k\sum_{r=1}^{j-1} p_r$, where k

* Corresponding author. Tel.: +1 305 348 3309; fax: +1 305 348 4126.
E-mail address: koulamas@fiu.edu (C. Koulamas).

0377-2217/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2007.08.015

is a normalizing constant and $\sum_{r=1}^{j-1} p_r$ denotes the total processing time of all already processed jobs on the machine prior to J_j, since in our single-machine environment a job's waiting-time is equal to the summation of the processing times of the already processed jobs. It is reasonable to assume that very long adjusted due dates will not be well-received by the customers and this can be prevented from happening by requiring that $k \leqslant 1$. Our scheduling objective is to minimize the maximum job lateness (to be formally defined in the next section). We show that for a given value of the due date normalizing constant k, an optimal sequence can be constructed in $O(n \log n)$ time (where n is the number of jobs) by implementing an index-based priority scheduling rule. We also show how to compute (in polynomial time) the optimal value of k, k^*, when the cost of assigning due dates is taken into account. Our approach for computing k^* is complicated by the fact that as k varies, the corresponding optimal sequence varies as well.

Our model can be easily implemented in a dynamic environment since the derived optimal job sequencing rule is an index-based rule based only on job characteristics. Newly arriving jobs can be inserted in the proper place in the sequence of the still unprocessed jobs without any major difficulty. Consequently, our model can be used for setting due dates in many practical operations in which due dates are initially negotiated with the customers and then adjusted accordingly to actual shop conditions. Such operations include small printing shops, auto repair shops, etc. in which the initially negotiated due dates with the customers may be adjusted based on actual waiting-times.

2. The optimal sequence for a fixed k value

We formally introduce our problem for a fixed k value as follows: There are n jobs J_j, $j = 1, \ldots, n$, (all available at time zero) to be scheduled non-preemptively on a continuously available single machine. Each job J_j has a processing time p_j and a baseline due date d_j. For any schedule (sequence) S of the n jobs, let $J_{[j]}$ denote the job in the jth position in S and let $C_{[j]}$ be the completion time of $J_{[j]}$ in S. The actual due date of $J_{[j]}$ in S, $d'_{[j]}$, is a linear function of its waiting-time $\sum_{r=1}^{j-1} p_{[r]}$. Specifically, $d'_{[j]} = d_{[j]} + k\sum_{r=1}^{j-1} p_{[r]}$, where $0 \leqslant k \leqslant 1$ is a normalizing constant. The lateness of job $J_{[j]}$ in S is defined as $L_{[j]} = C_{[j]} - d'_{[j]}$. Alternatively, L_j denotes the lateness of job J_j when it is preferable to indicate the job index j rather than its position $[j]$ (in that case, knowledge of the identity of all jobs preceding J_j in S is still needed to compute d'_j and $L_j = C_j - d'_j$, respectively). Let $P_{[j]} = \sum_{r=1}^{j} p_{[r]}$. Then

$$\begin{aligned} L_{[j]} &= C_{[j]} - \left(d_{[j]} + k \sum_{r=1}^{j-1} p_{[r]} \right) \\ &= \sum_{r=1}^{j} p_{[r]} - \left(d_{[j]} + k \sum_{r=1}^{j-1} p_{[r]} \right) \\ &= P_{[j]} - [d_{[j]} + k(P_{[j]} - p_{[j]})]. \end{aligned} \tag{1}$$

Define the maximum lateness for the schedule S as $L_{\max} = \max_{j=1,\ldots,n}\{L_{[j]}\}$. The following proposition shows that, for a given k, $L_{\max}$ can be minimized by using an index-based priority rule.

Proposition 1. *For any fixed k, $0 \leqslant k \leqslant 1$, the schedule obtained by sequencing the jobs in non-decreasing order of their $d_j - kp_j$ values minimizes L_{max}.*

Proof. By an adjacent pairwise job interchange argument. Suppose that $L_{\max}$ achieves its minimum for a given schedule S in which there are two adjacent jobs, J_j and J_i respectively, such that J_j immediately precedes J_i and with $d_i - kp_i < d_j - k\ p_j$. Consider schedule S' obtained by interchanging J_j and J_i in S so that J_i immediately precedes J_j in S'. We will show that $L_{\max}(S') \leqslant L_{\max}(S)$. Since the lateness values of all jobs other than J_j and J_i are the same in both S and S', the inequality $L_{\max}(S') \leqslant L_{\max}(S)$ is equivalent to $\max\{L_i(S'), L_j(S')\} \leqslant \max\{L_j(S), L_i(S)\}$, which, after some simplification, reduces to

$$\begin{aligned} &\max\{kp_i + kp_j - p_j - d_i, kp_j - d_j\} \\ &\quad \leqslant \max\{kp_i + kp_j - p_i - d_j, kp_i - d_i\}. \end{aligned} \tag{2}$$

Inequality (2) is true if the following two sufficient conditions hold: (C1) $kp_i + kp_j - p_j - d_i \leqslant k\ p_i - d_i$, and (C2) $kp_j - d_j \leqslant kp_i - d_i$. It is easy to see that (C1) holds if $k \leqslant 1$ and that (C2) holds if $d_i - k\ p_i \leqslant d_j - k\ p_j$. In view of the previous assumptions, both (C1) and (C2) hold and thus $L_{\max}(S') \leqslant L_{\max}(S)$. The repeated implementation of the above pairwise job interchange argument leads to an optimal schedule for minimizing $L_{\max}$ in which all jobs are ordered in the non-decreasing order of their $d_j - kp_j$ values. In principle, inequality (2) can be also satisfied by different sets of conditions stemming from comparing the terms in inequality (2) in alternative ways. However, all alternative ways

of comparing the terms in inequality (2) lead to either contradictory results or they require that $k \geqslant 1$ in order for (2) to be valid. Consequently, our index priority rule is the unique index rule satisfying inequality (2) under the assumption that $0 \leqslant k \leqslant 1$.

The optimal schedule in Proposition 1 is constructed in O($n$log$n$) time due to the required sorting operation. □

3. Assignment of the optimal due dates

We now turn our attention to the problem of assigning the optimal due dates. In order to do so, we must determine the optimal k value for a given cost function.

Define the parametric maximum lateness for schedule S as $L_{\max}(k)$ and observe that $L_{\max}(k)$ is a non-increasing function of k since each $d'_{[j]}$ is a non-decreasing function of k. This justifies the following total cost objective function

$$\mathrm{TC}(k) = \alpha k + L_{\max}(k), \tag{3}$$

where $\alpha > 0$ is a cost coefficient. The total cost in Eq. (3) is the sum of the cost of assigning long due dates and the cost of failing to meet assigned due dates. The due date assignment cost may represent the potential loss of customers good will due to quoting late due dates. The maximum lateness penalty cost is relevant especially when late deliveries result in various internal and external costs. As a result, the total cost expression in (3) represents the objective of the shop manager to strike a balance between assigning realistic due dates and at the same time reducing the risk of incurring late delivery penalties. Our objective is to determine the optimal value k^* (and the corresponding overall optimal schedule (sequence) S^*) which minimizes TC(k) for all $0 \leqslant k \leqslant 1$.

Proposition 1 can be used to construct the optimal schedule S^* in O($n$log$n$) time for a given k value. However, the optimal schedule S^* changes as the value of k changes. In order to address this complication, we first determine all critical k values in the $[0,1]$ range which induce an optimal schedule change.

For each pair of jobs J_i, J_j, $i = 1,\ldots,n-1$, $j = i+1,\ldots,n$, such that $p_i < p_j$ and $d_i < d_j$, define the quantity

$$z_{i,j} = (d_j - d_i)/(p_j - p_i) > 0. \tag{4}$$

The definition of $z_{i,j}$ facilitates the possible ordering of the quantities $d_i - k\,p_i$, $d_j - kp_j$ (and therefore of jobs J_i, J_j) for various k values. In the rest of the paper we use the notation $J_i \leftarrow J_j$ to indicate that J_i precedes J_j in an optimal schedule S^*. The next proposition summarizes some possible job orderings in S^*. Its proof is omitted since it is a straightforward implementation of (4) in conjunction with Proposition 1.

Proposition 2. *For any fixed* $0 \leqslant k \leqslant 1$,

(*A*1) *If* $p_i = p_j$ *and* $d_i < d_j$, *then* $d_i - kp_i \leqslant d_j - kp_j$ *and* $J_i \leftarrow J_j$.
(*A*2) *If* $p_i < p_j$, $d_i < d_j$, *and* $k \leqslant \min\{z_{i,j}, 1\}$, *then* $d_i - kp_i \leqslant d_j - kp_j$ *and* $J_i \leftarrow J_j$.
(*B*1) *If* $p_i \leqslant p_j$ *and* $d_i \geqslant d_j$, *then* $d_j - kp_j \leqslant d_i - k\,p_i$ *and* $J_j \leftarrow J_i$.
(*B*2) *If* $p_i < p_j$, $d_i < d_j$, *and* $z_{i,j} \leqslant k \leqslant 1$, *then* $d_j - kp_j \leqslant d_i - kp_i$ *and* $J_j \leftarrow J_i$.

Observe that conditions (*A*2) *and* (*B*2) *collectively correspond to the case where* $p_i < p_j$, $d_i < d_j$, *which is the only case not covered by conditions* (*A*1) *and* (*B*1). *Since* $z_{i,j} > 0$ *in this case, for any fixed* $0 \leqslant k \leqslant 1$ *value, either* $z_{i,j} \leqslant k \leqslant 1$ *which is covered by* (*B*2) *or* $k \leqslant z_{i,j}$ *which is covered by* (*A*2).

Conditions (*A*1) *and* (*B*1) *yield global job orderings in an optimal schedule* S^* *while conditions* (*A*2) *and* (*B*2) *yield local job orderings* (*dependent on the actual* $z_{i,j}$ *value*) *and can be used to divide the* $[0,1]$ *range into intervals so that for all k values in each interval the optimal schedule* S^* *remains the same. The requisite procedure for determining these intervals* (*and the corresponding optimal schedules*) *is formally presented next as Algorithm* A. *Algorithm* A *is based on the idea that if there are no* $z_{i,j}$ *values for any i, j pair in a certain subinterval of* $[0,1]$, *then a particular sequence determined by the conditions in Proposition* 2 *is optimal for all k values in that subinterval.*

Algorithm A

Step 1: For each job pair J_i, J_j, $i = 1,\ldots,n-1$, $j = i+1,\ldots,n$, such that $p_i < p_j$ and $d_i < d_j$, compute the corresponding $z_{i,j}$ value using (4).

Step 2: Arrange all $z_{i,j}$ values computed in Step 1 in non-decreasing order and eliminate any duplicate $z_{i,j}$ values as well as any $z_{i,j} \geqslant 1$ values. Let k_i, $i = 1,\ldots,m$, be the remaining

ordered $z_{i,j}$ values, where $0 < k_1 < k_2 < \cdots k_m < 1$.

Step 3: Let $k_0 = 0$ and $k_{m+1} = 1$. For each interval $[k_i, k_{i+1}]$, $i = 0, 1, \ldots, m$, compute its mid-point $\bar{k}_{i,i+1} = (k_i + k_{i+1})/2$.

Step 4: For each $\bar{k}_{i,i+1}$ value computed in Step 3, construct the optimal schedule $S^*_{i,i+1}$ by sequencing the jobs in non-decreasing order of their $d_j - \bar{k}_{i,i+1} p_j$ values.

END.

The complexity of Algorithm A is $O(n^2)$ determined by the number of pairwise job comparisons in Step 1.

We observed earlier that, for a given fixed schedule S, $L_{\max}(k)$ is a non-increasing function of k. Thus, $L_{\max}(k)$ is a non-increasing function of $k \in [k_i, k_{i+1}]$ since the same optimal schedule $S^*_{i,i+1}$ is used for all $k \in [k_i, k_{i+1}]$ values, $i = 0, 1, \ldots, m$. In order to find the optimal k value which minimizes (3) over all $k \in [0, 1]$ values, we first find the local optimal $k^*_{i,i+1}$ value, $i = 0, 1, \ldots, m$, for which $\mathrm{TC}(k^*_{i,i+1}) = \min_{k \in [k_i, k_{i+1}]} \mathrm{TC}(k)$. Then, the global optimal k^* can be determined by repeating the above procedure at most n^2 times (since there are no more than $n^2[k_i, k_{i+1}]$ intervals in the $[0, 1]$ range) so that

$$\mathrm{TC}(k^*) = \min\{\mathrm{TC}(k^*_{0,1}), \ldots, \mathrm{TC}(k^*_{m,m+1})\}. \tag{5}$$

Consider the optimal schedule $S^*_{i,i+1}$ corresponding to the minimum $L_{\max}(k)$ for each $k \in [k_i, k_{i+1}]$, $i = 0, 1, \ldots, m$. In order to simplify exposition somewhat, we appropriately renumber the jobs so that we may assume that $S^*_{i,i+1}$ corresponds to the sequence $(J_1, \ldots, J_n)$. In view of (1), since job J_r is in position r in the $(J_1, \ldots, J_n)$ sequence and $L_{[r]}$ denotes the lateness of the job in position r, $r = 1, \ldots, n$,

$$\begin{aligned} L_{[r]} &= P_{[r]} - (d_{[r]} + k(P_{[r]} - p_{[r]})) \\ &= P_{[r]} - d_{[r]} - k(P_{[r]} - p_{[r]}) \\ &= \sum_{s=1}^{r} p_s - d_r - k\left(\sum_{s=1}^{r} p_s - p_r\right). \end{aligned}$$

Consequently, $L_{[r]}$ can be viewed as a linear function of k with slope $-b_r$ and intercept a_r where $a_r = \sum_{s=1}^{r} p_s - d_r$ and $b_r = \sum_{s=1}^{r} p_s - p_r$ respectively, $r = 1, \ldots, n$. Therefore,

$$\begin{aligned} L_{\max}(k) &= \max\{L_{[1]}, \ldots, L_{[n]}\} \\ &= \max\{a_1 - b_1 k, \ldots, a_n - b_n k\}, \end{aligned} \tag{6}$$

where $k \in [k_i, k_{i+1}]$. According to (6), $L_{\max}(k)$ is a piecewise linear convex function in the interval $[k_i, k_{i+1}]$, since it is a maximum of n linear functions. Furthermore, $L_{\max}(k)$ is non-increasing because all slope values, $-b_1, \ldots, -b_n$, are non-positive.

Geometrically, the $L_{\max}(k)$ function in the $[k_i, k_{i+1}]$ interval consists of the convex upper envelope (maximum) of some c linear segments selected from the n linear segments in (6). The identification of these c linear segments is the objective of Algorithm B which determines the "non-inferior" linear segments among the n segments defined by the n terms under the maximum of the piecewise linear $L_{\max}(k)$ function in the $[k_i, k_{i+1}]$ interval.

Algorithm B

Step 1: For a fixed i, let $S^*_{i,i+1}$ denote the optimal schedule corresponding to the sequence $(J_1, \ldots, J_n)$. Compute $a_r = \sum_{s=1}^{r} p_s - d_r$, $b_r = \sum_{s=1}^{r} p_s - p_r$ for $r = 1, \ldots, n$.

Step 2: Reorder the $\{a_r, b_r\}$ values computed in Step 1 so that $a_1 \leqslant \cdots \leqslant a_n$ (and $b_r \geqslant b_{r+1}$ in case of $a_r = a_{r+1}$). If for any r, $b_r \geqslant b_{r+1}$ in the reordered sequence, then eliminate $\{a_{r+1}, b_{r+1}\}$. Assume that there are $n' \leqslant n$ linear segments $L_{[r]}$ such that $a_1 \leqslant \cdots \leqslant a_{n'}$ and $b_1 < \cdots < b_{n'}$.

Step 3: For each r, $r = 1, \ldots, n'$, calculate the y-coordinate of the linear segment $L_{[r]}$ at k_i as $y_r = a_r - k_i b_r$.

Step 4: For each pair of linear segments $a_u - b_u k$, $a_v - b_v k$, $u = 1, \ldots, n' - 1$, $v = u + 1, \ldots, n'$, compute the x-coordinate of their intersection point $x_{v,u}$ as $x_{v,u} = (a_v - a_u)/(b_v - b_u) \geqslant 0$.

Step 5: Select $r_1 \in \{1, \ldots, n'\}$ such that $y_{r_1} = \max_{r=1,\ldots,n'}\{y_r\}$; in case of ties, select the smallest such r_1 value. Let $x_{r_0, r_1} = k_i$.

Step 6: For $s = 2, \ldots, n'$, select $r_s \in \{1, \ldots, r_{s-1} - 1\}$ such that $x_{r_{s-1}, r_s} = \min_{r \in \{1,\ldots,r_{s-1}-1\}}\{x_{r_{s-1},r} : x_{r_{s-1},r} \in [x_{r_{s-2},r_{s-1}}, k_{i+1})\}$; in case of ties, select the smallest such r_s value. If x_{r_{s-1},r_s} does not exist, then go to Step 7 with $c = s - 1$; else, $c = s$, next s.

Step 7: The $L_{\max}(k)$ function for $k \in [k_i, k_{i+1}]$ consists of c linear segments given as $a_{r_w} - b_{r_w} k$ for $k \in [x_{r_{w-1}, r_w}, x_{r_w, r_{w+1}}]$, $w = 1, \ldots, c - 1$, and $a_{r_c} - b_{r_c} k$ for $k \in [x_{r_{c-1}, r_c}, k_{i+1}]$, respectively.

END.

The complexity of Algorithm B is $O(n^2)$ determined by the number of pairwise job comparisons in Step 4.

Since $L_{\max}(k)$ is a piecewise linear convex function in the $[k_i, k_{i+1}]$ interval, then $TC(k)$ is a piecewise linear convex function in that interval as well. Thus, any local minimum $k^*_{i,i+1}$ of $TC(k)$ is also a global minimum in the $[k_i, k_{i+1}]$ interval and it can be computed by Procedure C detailed next. Procedure C balances the due date assignment cost (expressed by αk) with the lateness penalty cost $L_{\max}(k)$ in the $[k_i, k_{i+1}]$ interval and determines the minimum $k^*_{i,i+1}$ value of $TC(k)$ in the $[k_i, k_{i+1}]$ interval based on comparisons of the slope α of the linear function αk and the slopes of the non-inferior linear segments of $L_{\max}(k)$ determined by Algorithm B.

Procedure C

(a) If $b_{r_1} > \cdots > b_{r_c} \geqslant \alpha$, $TC(k)$ is non-increasing in the $[k_i, k_{i+1}]$ interval and $k^*_{i,i+1} = k_{i+1}$.
(b) If $\alpha \geqslant b_{r_1} > \cdots > b_{r_c}$, $TC(k)$ is non-decreasing in the $[k_i, k_{i+1}]$ interval and $k^*_{i,i+1} = k_i$.
(c) If $b_{r_s} \geqslant \alpha \geqslant b_{r_{s+1}}$ for some $s = 1, \ldots, c-1$, then $\alpha - b_{r_s} \leqslant 0$ and $\alpha - b_{r_{s+1}} \geqslant 0$, or equivalently, $TC(k)$ is non-increasing in the $[k_i, x_{r_s,r_{s+1}}]$ interval and non-decreasing in the $[x_{r_s,r_{s+1}}, k_{i+1}]$ interval, where $x_{r_s,r_{s+1}} = (a_{r_s} - a_{r_{s+1}})/(b_{r_s} - b_{r_{s+1}}) > 0$ is the x-coordinate of the intersection point of the lines $a_{r_s} - b_{r_s}k$ and $a_{r_{s+1}} - b_{r_{s+1}}k$, respectively; therefore, $k^*_{i,i+1} = x_{r_s,r_{s+1}}$.

END.

The complexity of Procedure C is $O(n)$. The condition in (c) is always true for some s value when neither the condition in (a) nor the condition in (b) are true due to the way $L_{\max}(k)$ is constructed in Step 7 of Algorithm B; also, $c \geqslant 2$ in that case. This ensures that Procedure C always succeeds in computing $k^*_{i,i+1}$.

Algorithm B and Procedure C are both implemented for each $[k_i, k_{i+1}]$ interval to determine the corresponding minimum $k^*_{i,i+1}$ value. These $k^*_{i,i+1}$ minima are then used to determine the overall global minimum k^* of $TC(k)$ on the $[0,1]$ interval according to (5). Since there are at most $n^2[k_i, k_{i+1}]$ intervals, the complexity of the overall optimization procedure is $O(n^4)$.

The optimal job due dates d'^*_j, $j = 1, \ldots, n$, can now be computed as follows: first, the jobs are sequenced according to the optimal schedule $S^*_{i,i+1}$ for the specific $[k_i, k_{i+1}]$ interval which supplied the overall optimal k^* value. Then, d'^*_j, $j = 1, \ldots, n$, are computed using $k = k^*_{i,i+1}$ and the $p_{[r]}$ values, $r = 1, \ldots, j-1$, according to the optimal $S^*_{i,i+1}$ schedule.

We close by pointing out that if the linear due date assignment cost αk is replaced by $\alpha(k)$ in (3), where $\alpha(k)$ is a non-decreasing convex function, then $TC(k)$ is no longer a piecewise linear convex function in each $[k_i, k_{i+1}]$ interval and thus the polynomial time Procedure C is no longer applicable. Instead, $TC(k)$ is now a convex function and a global minimum of this function has to be computed separately in each $[k_i, k_{i+1}]$ interval. This new problem is more difficult to solve and it requires a more elaborate solution approach.

4. Conclusions

We considered the problem of determining optimal due dates when due dates are linear functions of the job waiting-times in a single-machine setting under the $L_{\max}$ objective. We showed that an optimal sequence can be determined in $O(n \log n)$ time by implementing an index-based priority rule for a fixed value of the due date normalizing constant k. We showed how to determine the global optimal k^* value in polynomial time which in turn led to the determination of the optimal due dates as linear functions of k^*. Our results pave the way for pursuing analytical results with waiting-times-based job due dates in more general shop environments.

Acknowledgement

We would like to thank two anonymous referees for their constructive criticism which helped us improve an earlier version of the paper.

References

Cheng, T.C.E., Gupta, M.C., 1989. Survey of scheduling research involving due date determination decisions. European Journal of Operational Research 38, 156–166.
Eilon, S., Chowdhury, I.G., 1976. Due-date in job shop scheduling. International Journal of Production Research 14, 223–238.
Ragatz, G.L., Mabert, V.A., 1985. A simulation analysis of due-date assignment rules. Journal of Operations Management 5, 27–39.
Weeks, J.K., 1979. A simulation study of predictable due-dates. Management Science 25, 363–373.