On the fixed interval due-date scheduling problem

Chung-Yee Lee\textsuperscript{a}, Chung-Lun Li\textsuperscript{b}

\textsuperscript{a} Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611, USA
\textsuperscript{b} John M. Olin School of Business, Washington University, St. Louis, MO 63130, USA

Received 27 October 1993; revised 29 November 1994

Abstract

We consider the nonpreemptive single machine scheduling problem with multiple due-dates (delivery dates), where the time between two consecutive due-dates is a constant. Given a set of jobs, we are interested in scheduling the jobs such that the sum of the total due-date cost and the total earliness cost is minimized with the constraint that each job must be finished at or before its due-date. We show that this problem is strongly NP-hard in general. We then analyze the problem where the number of due-dates is bounded by a given constant. We describe a pseudo-polynomial dynamic programming algorithm for this restricted problem. A heuristic is also provided and worst-case analysis is performed. An efficient algorithm is developed to solve the special case where all the job processing times are identical.

Keywords: Delivery dates; Scheduling; Single machine

1. Introduction

The significance of assigning accurate due-dates to jobs and having them meet the due-dates is well-addressed in the scheduling literature. In recent years, there appeared a number of articles on machine scheduling problems with earliness/tardiness penalties about a single common due-date (see [4]). However, in many manufacturing environments, there are multiple delivery dates in the planning horizon while the time interval between any two consecutive delivery dates is constant. For example, a company may deliver only once a week, and the delivery dates are scheduled on every Friday. As mentioned in [9], this kind of fixed interval delivery strategy results from transportation and handling economies that make it advantageous to consolidate shipments.

In this paper, we consider the nonpreemptive single machine scheduling problem with multiple due-dates (delivery dates), where the time between two consecutive due-dates is a given constant $\tau$. Given a set of $n$ simultaneously available jobs, we are interested in scheduling the jobs such that the sum of the total due-date cost and the total earliness cost is minimized with the constraint that each job must be finished at
or before its due-date. We assume that the due-date cost of each job is proportional to
the length of its due-date, and the earliness cost of a job is proportional to the amount
of earliness. Since all due-dates are assumed to be integer multiple of a constant, this
problem is called the fixed interval due-date scheduling problem with earliness and
due-date costs and is denoted by P. We also denote the problem by P(m) if the due-
dates are restricted to $\tau, 2\tau, \ldots, m\tau$, i.e., if all the jobs are required to be completed by
time $m\tau$.

Chhajed [6] has considered problem P(2), i.e., the two-interval due-date scheduling
problem with earliness and due-date costs. He has shown that P(2) is NP-hard and
developed a procedure to obtain lower and upper bounds on the optimal solution value.
Lee et al. [8] have studied the common due-date problem with earliness-tardiness cost
and number of tardy job cost, where the number of tardy job cost can be considered
implicitly as the due-date cost. Matsuo [9] has considered a problem similar to ours.
He studied the problem with fixed delivery dates to minimize the sum of overtime
and weighted tardiness costs where each job has a different deadline date. He showed
that the problem in its simplest form is NP-hard. He then presented an approximation
algorithm based on a capacitated transshipment formulation. Other machine scheduling
models with earliness costs and deadline constraints have also been considered by
Schneeberger [11], Ahmadi and Bagchi [1-3], and Chand and Schneeberger [5].

In this paper, we show that problem P is NP-hard in the strong sense, while problem
P(m) is pseudo-polynomial time solvable if $m$ is a fixed number. We then provide an
efficient heuristic to solve problem P(m). This heuristic has a constant worst-case error
bound provided that $m$ is given as a constant. An $O(n^3)$ time algorithm is developed
to solve the special case where all the job processing times are identical.

2. Notation and properties of the optimal solution

Let $\{J_1, \ldots, J_n\}$ be the set of all jobs and $p_j$ be the processing time of job $J_j$
($j = 1, \ldots, n$). As mentioned in Section 1, parameter $\tau$ is the length of each due-
date interval. Thus, the set of all possible due-dates is $\{\tau, 2\tau, 3\tau, \ldots\}$. Let $C_j$ be the
completion time of job $J_j$. If we have a schedule with $(i-1)\tau < C_j \leq i\tau$, then we
know that the due-date of $J_j$ cannot be less than $i\tau$. Also, it is obvious that it is
nonoptimal to assign a due-date greater than $i\tau$ to this job. Hence, the due-dates of the
jobs are determined once a schedule is provided. In other words, our problem is to
form a schedule of jobs that minimizes the total cost, rather than solving a "due-date
assignment problem". Let $\beta$ be the due-date cost per due-date interval and $\alpha$ be the
earliness penalty (or inventory holding cost) per unit time. Thus, if $(i-1)\tau < C_j \leq i\tau$,
then the due-date cost of $J_j$ is $i\beta$ and the earliness penalty of $J_j$ is $(i\tau - C_j)\alpha$. In this
case, job $J_j$ will be delivered at time $i\tau$. Here, we assume that $\beta$ is independent of $i$
and $j$.

Throughout this paper, we assume that $p_j \leq \tau$ for every job $J_j$ and that all the jobs
are available at time zero. Idle time between jobs is allowed while preemption is not
allowed. Without loss of generality, we also assume that the per unit earliness penalty is equal to one (i.e., \(\alpha = 1\)). In problem \(P(m)\), we further assume that \(2 \leq m \leq n\), and that \(\sum_{j=1}^{n} p_j \leq m\tau\) so that a feasible schedule always exists. We now state some important properties of the optimal solution of problems \(P\) and \(P(m)\).

**Property 1.** There exists an optimal solution such that all jobs having the same due-date are sequenced in the Longest Processing Time (LPT) order.

**Proof.** By a simple job interchange argument. \(\square\)

**Property 2.** There exists an optimal solution such that if a job \(J_j\) completes at time \(t\) and there is an idle time after \(t\), then \(t = k\tau\) for some \(k \in \mathbb{Z}^+\).

**Proof.** For otherwise we can shift \(J_j\) to the right and reduce the total cost. \(\square\)

**Property 3.** There exists an optimal solution such that jobs are divided into several groups, where the jobs in each group are processed consecutively and the last job in the group completes at time \(k\tau\) for some \(k \in \mathbb{Z}^+\).

**Proof.** It follows directly from Property 2. \(\square\)

For example, in Fig. 1, \(J_1\) and \(J_2\) belong to the 1st group, and \(J_3, J_4\) and \(J_5\) belong to the 2nd group. In this case, we say that group 1 “covers” the 1st due-date period, while group 2 “covers” the 2nd and 3rd due-date periods. The due-date of \(J_1\) and \(J_2\) is \(\tau\), the due-date of \(J_3\) is \(2\tau\), and the due-date of \(J_4\) and \(J_5\) is \(3\tau\).

**Remark 1.** If \(\beta = 0\) and \(\sum_{j=1}^{n} p_j\) is significantly small (e.g., \(\sum_{j=1}^{n} p_j \leq \tau\)) then problem \(P(m)\) is equivalent to the \(Pm||\sum C_j\) problem (i.e., the \(n\)-job \(m\)-parallel-machine scheduling problem with an objective of minimizing the total completion time; see Pinedo [10] for a detailed discussion of this problem). In fact, for any instance of \(P(m)\) with \(\beta = 0\) and \(\sum_{j=1}^{n} p_j\) small enough, the optimal solution value of \(P(m)\) is equal to \(Z - \sum_{j=1}^{n} p_j\), where \(Z\) is the optimal solution value of the corresponding instance of \(Pm||\sum C_j\).

3. Computational complexity

In this section, we show that our problem is strongly NP-hard in general. A pseudopolynomial time algorithm is then provided for the case when the number of intervals is fixed. We first consider the computational complexity of problem \(P\).
Theorem 1. Problem P is NP-hard in the strong sense.

Proof. We transform the Numerical 3-Dimensional Matching (N3DM) problem to problem P. An instance of N3DM has three disjoint sets $W = \{a_{11}, \ldots, a_{1m}\}$, $X = \{a_{21}, \ldots, a_{2m}\}$, and $Y = \{a_{31}, \ldots, a_{3m}\}$ of positive integers with $|W| = |X| = |Y| = m$ and a bound $B \in \mathbb{Z}^+$. (We may assume that $a_{kj} < B$ for all $k = 1, 2, 3$ and $i = 1, \ldots, m$.) The instance asks whether $W \cup X \cup Y$ can be partitioned into $m$ disjoint sets $A_1, A_2, \ldots, A_m$ such that each $A_i$ contains exactly one element from each of $W$, $X$, and $Y$ and such that, for $1 \leq i \leq m$, $\sum_{A_i, a = B}$. N3DM is known to be NP-complete in the strong sense (see [7]).

Given an arbitrary instance of N3DM, we let

$$M = 3 \sum_{i=1}^{m} a_{1i} + 2 \sum_{i=1}^{m} a_{2i} + \sum_{i=1}^{m} a_{3i} + (24m + 4)mB$$

and

$$K = (2m^2 + 2m + 1)M.$$

Then we construct the following instance of P with $4m + K$ jobs:

- $p_i = K$, for $i = 1, \ldots, m$;
- $p_{m+i} = a_{1i} + 4mB$, for $i = 1, \ldots, m$;
- $p_{2m+i} = a_{2i} + (4m + 1)B$, for $i = 1, \ldots, m$;
- $p_{3m+i} = a_{3i} + (4m + 2)B$, for $i = 1, \ldots, m$;
- $p_{4m+i} = \tau$, for $i = 1, \ldots, K$;
- length of each due-date interval, $\tau = K + (12m + 4)B$;
- due-date cost, $\beta = M$;
- threshold, $L = K + MK \left( m + \frac{K + 1}{2} \right)$.

Obviously, this construction can be done in pseudo-polynomial time. Suppose there exists a partition $\{A_1, A_2, \ldots, A_m\}$ of $W \cup X \cup Y$ such that each $A_i$ contains exactly one element from each of $W$, $X$, and $Y$ and $\sum_{A_i, a = B}$ for $i = 1, \ldots, m$. Then for $i = 1, \ldots, m$, we assign due-date $i\tau$ to the jobs

$$\{J_i \cup \{J_{m+k} \mid a_{1k} \in A_i\} \cup \{J_{2m+k} \mid a_{2k} \in A_i\} \cup \{J_{3m+k} \mid a_{3k} \in A_i\}$$

and arrange these 4 jobs in the Longest Processing Time (LPT) order. We schedule the jobs $J_{4m+1}, \ldots, J_{4m+K}$ to complete at time $(m+1)\tau, \ldots, (m+K)\tau$, respectively. The total due-date cost of this assignment is

$$4(\beta + 2\beta + \cdots + m\beta) + (m+1)\beta + \cdots + (m+K)\beta$$

$$= 2m(m+1)M + MK \left( m + \frac{K + 1}{2} \right).$$

Note that for $i = 1, \ldots, m$, the total processing time of the jobs get assigned due-date $i\tau$ is $p_i + \sum_{A_i, a = B}$, $(4m + 1)B + (4m + 2)B = K + (12m + 4)B = \tau$ (see
Fig. 2). Hence, the total earliness of jobs $J_{2m+1}, \ldots, J_{4m}$ and $J_{4m+1}, \ldots, J_{4m+K}$ is 0; the total earliness of jobs $J_{2m+1}, \ldots, J_{3m}$ is $\sum_{i=1}^{m} a_{1i} + 4m^2B$; the total earliness of jobs $J_{3m+1}, \ldots, J_{4m}$ is $\sum_{i=1}^{m} a_{1i} + \sum_{i=1}^{m} a_{2i} + (8m + 1)mB$; and the total earliness of jobs $J_1, \ldots, J_m$ is $\sum_{i=1}^{m} a_{1i} + \sum_{i=1}^{m} a_{2i} + \sum_{i=1}^{m} a_{3i} + (12m + 3)mB$. Therefore, the total cost of this schedule is

$$2m(m+1)M + MK \left(m + \frac{K+1}{2}\right) + \sum_{i=1}^{m} a_{1i} + \sum_{i=1}^{m} a_{2i} + (24m + 4)mB$$

Conversely, suppose there exists a solution to problem P with total cost at most $L$. First note that since no job has processing time greater than $\tau$ and the total processing time of all jobs is equal to $(m + K)\tau$, the total due-date cost for those jobs with due-dates $(m + 1)\tau, (m + 2)\tau, \ldots, (m + K)\tau$ is at least $[(m + 1) + (m + 2) + \cdots + (m + K)]B = MK(m + (K+1)/2)$. Thus, no two jobs from $J_1, \ldots, J_m$ can have the same due-date, for otherwise one of these two jobs will have an earliness of at least $K$ and the total cost will be greater than $K + MK(m + (K+1)/2) = L$, which is impossible. Note also that all the jobs must be completed by time $(m + K)\tau$, for otherwise the total due-date cost of the last $K$ jobs will exceed $L$. Hence, there must be no idle time between jobs in the schedule, and jobs $J_1, \ldots, J_m$ and $J_{4m+1}, \ldots, J_{4m+K}$ must be assigned different due-dates (recall that the processing time of each $J_{4m+1}, \ldots, J_{4m+K}$ is equal to $\tau$). By Property 1, each of these $m + K$ jobs must be the first job in each “due-date period”. Thus, there are at most four jobs with due-date $\tau$, because the total processing time of any four small jobs is greater than $\tau - K$ (here, “small” jobs refer to jobs $J_{m+1}, \ldots, J_{4m}$). Similarly, there are at most $4i$ jobs with due-date less than or equal to $i\tau$, for $i = 1, \ldots, m$. Since there are $4m + K$ jobs to be arranged into $m + K$ periods and at most $4i$ jobs can be scheduled within the first $i$ periods, the least possible total due-date cost will be achieved if we schedule four jobs in each of the first $m$ periods, and one job in each of periods $m + 1, m + 2, \ldots, m + K$. The total due-date cost of such a schedule is $MK(m + (K+1)/2) + 4(1 + \cdots + m)M = L - M$. 

Fig. 2. The schedule in the proof of Theorem 1.
This implies that the total earliness cost of all the jobs is at most $M$. The minimum total earliness of a schedule with jobs $J_1, \ldots, J_{4m}$ scheduled in $m$ different periods is at least equal to the minimum total flow time of assigning $\{J_{m+1}, J_{m+2}, \ldots, J_{4m}\}$ onto $m$ parallel machines (see Remark 1). The optimal solution to this $Pm||\sum C_j$ problem is obtained by the SPT rule (see [10]), and the optimal total flow time is equal to $\sum_{i=1}^{m} [a_{3i} + (4m+2)B] + 2 \sum_{i=1}^{m} [a_{2i} + (4m+1)B] + 3 \sum_{i=1}^{m} [a_{i} + 4mB] = M$. Hence, the minimum total earliness of our schedule is equal to $M$, which is achieved if each of $J_{m+1}, J_{m+2}, \ldots, J_{2m}$ completes at its due-date and each of $J_{m+1}, J_{m+2}, \ldots, J_{2m}$ is preceded by a job in $\{J_{2m+1}, \ldots, J_{3m}\}$ and each of $J_{2m+1}, J_{2m+2}, \ldots, J_{3m}$ is preceded by a job in $\{J_{3m+1}, \ldots, J_{4m}\}$ (since the processing times of $J_{3m+1}, \ldots, J_{4m}$ are greater than those of $J_{2m+1}, \ldots, J_{3m}$, which in turn are greater than those of $J_{m+1}, \ldots, J_{2m}$). In other words, exactly one job from each of $\{J_{m+1}, \ldots, J_{2m}\}$, $\{J_{2m+1}, \ldots, J_{3m}\}$, and $\{J_{3m+1}, \ldots, J_{4m}\}$ must be assigned to the $i$th period and the total processing time of the three small jobs with the same due-date must be equal to $(12m + 4)B$. This implies the existence of a partition $A_1, A_2, \ldots, A_m$ of $W \cup X \cup Y$ such that each $A_i$ contains exactly one element from each of $W$, $X$, and $Y$, and $\sum A_i = B$ for $i = 1, \ldots, m$.

This completes the proof of the theorem. □

We now consider problem $P(m)$. If $m$ is a fixed number, $P(m)$ can be solved in pseudo-polynomial time by using the following dynamic program. We first sort the jobs in nondecreasing processing times, i.e., $p_1 \leq p_2 \leq \cdots \leq p_n$.

**Dynamic program for $P(m)$**

1. Define $f(n_1, \ldots, n_m; t_1, \ldots, t_m) =$ minimum cost to schedule jobs $J_1, J_2, \ldots, J_{n_1+\cdots+n_m}$ such that the first $n_1$ jobs $X_1 = \{J_1, \ldots, J_{n_1}\}$ have due-date $\tau$, the next $n_2$ jobs $X_2 = \{J_{n_1+1}, \ldots, J_{n_1+n_2}\}$ have due-date $2\tau$, etc., and such that the total processing time of the jobs in $X_i$ is $t_i$ ($i = 1, \ldots, m$).

2. Recurrence relation:

$$f(n_1, \ldots, n_m; t_1, \ldots, t_m) = \min_{s=1, \ldots, m} \{ f(n_1, \ldots, n_{i-1}, n_i-1, n_{i+1}, \ldots, n_m; t_1, \ldots, t_{i-1}, t_{i} - \hat{\rho}, t_{i+1}, \ldots, t_m) + (t_i - \hat{\rho} + \delta_i) + i\beta + F_i \}$$

where

$$\hat{\rho} = p_{n_1+\cdots+n_m}$$

and

$$\delta_i = \max \{(t_{i+1} - \tau)^+, (t_{i+1} + t_{i+2} - 2\tau)^+, \ldots, (t_{i+1} + \cdots + t_m - (m - i)\tau)^+ \}$$
and

\[ F_i = \begin{cases} +\infty, & \text{if } (t_i + \delta_i - \tau)^+ - [(i - 1)\tau - \sum_{j=1}^{i-1} t_j]^+ > 0; \\ \sum_{k=1}^{i-1} n_{i-k} [(t_i + \delta_i - \tau)^+ - [(k-1)\tau - \sum_{j=k+1}^{i-1} t_j]^+]^+, & \text{otherwise}; \end{cases} \]

and \((x)^+ \equiv \max\{x, 0\}\). [Note: \(t_m\) has to be equal to \(\sum_{j=1}^{n_m} p_j - \sum_{i=1}^{m-1} t_k\].

(3) Boundary condition

\[ f(0, \ldots, 0; 0, \ldots, 0) = 0. \]

(4) Objective

\[ \min \left\{ f(n_1, \ldots, n_m; t_1, \ldots, t_m) \left| n_1 + \cdots + n_m = n \text{ and } t_1 + \cdots + t_m = \sum_{j=1}^{n} p_j \right. \right\}. \]

In the above recurrence relation, we want to select the best possible due-date period for the \((n_1 + \cdots + n_m)\)th job. If this job is assigned to the \(i\)th due-date period, then the total cost of the partial schedule containing jobs \(J_1, \ldots, J_{n_1+\cdots+n_m}\) is equal to

\[ f(n_1, \ldots, n_{i-1}, n_i - 1, n_{i+1}, \ldots, n_m; t_1, \ldots, t_{i-1}, t_i - \hat{\beta}, t_{i+1}, \ldots, t_m) \]

plus the cost incurred by assigning \(J_{n_1+\cdots+n_m}\) to the schedule. Note that in the recurrence relation, \((t_i - \hat{\beta} + \delta_i)\) is the earliness cost and \(i\beta\) is the due-date cost of \(J_{n_1+\cdots+n_m}\), where \(\delta_i\) is the earliness of the last job in \(X_i\). \(F_i\) is the additional earliness cost of the jobs with due-dates \(\tau, 2\tau, \ldots, (i-1)\tau\) incurred by assigning \(J_{n_1+\cdots+n_m}\) to the \(i\)th due-date period. Note that for \(1 \leq k < i\), \([(t_i + \delta_i - \tau)^+ - [(k-1)\tau - \sum_{j=k+1}^{i-1} t_j]^+]^+\) is the increase in earliness of each job with due-date \((i - k)\tau\) if \(J_{p_1+\cdots+n_m}\) is assigned due-date \(i\tau\). If \((t_i + \delta_i - \tau)^+ - [(i-1)\tau - \sum_{j=1}^{i-1} t_j]^+ > 0\), then infeasibility will occur if \(J_{n_1+\cdots+n_m}\) is assigned to the \(i\)th due-date period. For example, we consider the schedule depicted in Fig. 3(i) with \(m = 4\) and \(\tau = 10\). Six jobs have already been scheduled. Suppose the seventh job with 7 units of processing time is assigned due-date 30, then as shown in Fig. 3(ii), the earliness of \(J_5, J_1, J_3, J_2\) increases by 1, 1, 3, 3 units, respectively. In this case, \(F_3 = \sum_{k=1}^{2} n_{3-k}[(t_3-10)^+ - [(k-1)10 - \sum_{j=4-k}^{2} t_j]^+]^+ = 8\) (note: \(n_1 = 2, n_2 = 2, t_1 = 8, t_2 = 8, t_3 = 13\)). Now, suppose the processing time of the seventh job is 9, then as shown in Fig. 3(iii), the solution becomes infeasible if \(J_7\) is assigned due-date 30. In this case, \((t_3-10)^+ - \sum_{j=1}^{2} (10-t_{3-j})^+ > 0\), and \(F_3 = +\infty\) (note: \(t_1 = 15\)).

Note that in the recurrence relation, \(t_m\) has to be equal to \(\sum_{j=1}^{n_m} p_j - \sum_{i=1}^{m-1} t_k\), i.e., the value of \(t_m\) is determined by the values of \(t_1, \ldots, t_{m-1}\). Hence, the total number of feasible combinations of \(n_1, \ldots, n_m; t_1, \ldots, t_m\) is bounded by \(n^m(2\tau)^{m-1}\), while each \(f(n_1, \ldots, n_m; t_1, \ldots, t_m)\) can be evaluated in constant time if \(m\) is constant. Hence, the
computational complexity of this dynamic program is $O(n^m \tau^{m-1})$ provided that $m$ is a fixed number.

**Remark 2.** Consider the special situation as depicted in Fig. 4(i), we have $X_1 = \{J_1, J_2\}$, $X_2 = \{J_3, J_4, J_5, J_6\}$, and $X_3 = \{J_7\}$. If a new job $J_6$ is added to $X_3$, then the completion time of $J_6$ will be less than $\tau$ (see Fig. 4(ii)). In this case, we still treat $J_6$ as an element of $X_2$ with due-date $2\tau$ in the dynamic program. Since $f(2, 4, 2; 5, 12, 12)$ is obviously greater than $f(3, 3, 2; 8, 9, 12)$ in this example, the partial schedule $(n_1, n_2, n_3; t_1, t_2, t_3) = (2, 4, 2; 5, 12, 12)$ will not result in optimal solution. Hence, treating $J_6$ as an element of $X_2$ will not affect the validity of the optimal solution.

**4. Heuristics and worst-case analysis**

We now present a heuristic for problem $P(m)$ and analyze its worst-case performance. We first renumber the jobs such that $p_1 \leq p_2 \leq \cdots \leq p_n$.

**Heuristic H for P(m)**

*Step 1:* Arrange the jobs in LPT order (i.e., $(J_n, J_{n-1}, \ldots, J_2, J_1)$) and process them as late as possible, i.e., the last job, $J_1$, will be completed at time $m\tau$ and no idle time is inserted between any two consecutive jobs.
Step 2: For \( i = 1, \ldots, m-1 \), if no job is currently assigned due-date \( i \tau \), then move the job \( J_{i-1} \) to the \( i \)th due-date period and have this job complete at time \( i \tau \), otherwise STOP.

Note that in the \( i \)th iteration of Step 2, we only move one job to the \( i \)th due-date period if no job is currently assigned to this period. The computational complexity of this heuristic is \( O(n \log n) \). Let \( Z_H \) denote the solution value obtained by the heuristic and \( Z^* \) denote the corresponding optimal solution value; let \( Z^H_e \) and \( Z^H_d \) be the total earliness penalty and total due-date cost, respectively, of the jobs in the heuristic solution; and let \( Z^*_e \) and \( Z^*_d \) be the total earliness penalty and total due-date cost, respectively, of the jobs in the optimal solution.

**Theorem 2.** Heuristic \( H \) has a worst-case error bound of \( Z^H/Z^* \leq m \).

**Proof.** We first consider a modified problem \( P'(m) \) with the same jobs as in problem \( P(m) \) but with a new due-date cost \( b' = 0 \) and a new due-date interval \( \tau' = \sum_{i=1}^n p_i \). To simplify the analysis, we assume that in this modified problem, \( n = km \), where \( k \) is an integer (since we can always put in additional jobs with zero processing time without changing the optimal solution of \( P'(m) \)). Obviously, the optimal solution value of \( P'(m) \) is a lower bound of \( Z^*_e \). Moreover, solving \( P'(m) \) is equivalent to solving the \( Pm||C_j \) scheduling problem (see Remark 1) by forming an SPT schedule on the \( m \) parallel processors. That is, if we assign jobs \( J_i, J_{m+i}, J_{2m+i}, \ldots, J_{(k-1)m+i} \) to due-date \( i \tau \) (\( i = 1, \ldots, m \)) and arrange the jobs in each due-date period in LPT order, then the schedule is optimal to \( P'(m) \). This optimal schedule has a total cost...
of

\[(k - 1)(p_1 + \cdots + p_m) + (k - 2)(p_{m+1} + \cdots + p_{2m}) + \cdots + 2(p_{(k-3)m+1} + \cdots + p_{km}) = \sum_{i=1}^{k-1} (k-i) \sum_{j=1}^{m} p_{(i-1)m+j}.\]

Hence,

\[Z^*_c \geq \sum_{i=1}^{k-1} (k-i) \sum_{j=1}^{m} p_{(i-1)m+j}. \tag{1}\]

We now consider the heuristic solution of \(P(m)\). Let \(r\) be the number of jobs moved during Step 2 of heuristic \(H\) \((0 \leq r \leq m-1)\). Clearly,

\[p_1 + p_2 + \cdots + p_{n-r-1} > (m-r-1)\tau\]

(otherwise, in the heuristic solution \(J_{n-r}\) completes after \((r+1)\tau\) and no job is assigned due-date \((r+1)\tau\), which is impossible). This, together with the fact that \(p_j + \cdots + p_{n-r-1} \leq (n-r-j)\tau\), implies that

\[p_1 + p_2 + \cdots + p_{j-1} > (j-n+m-1)\tau,\]

for \(j = n-m+2, \ldots, n-r\). For \(j = 1, \ldots, n\), let \(E_j\) be the earliness of \(J_j\) in the heuristic solution. Thus,

\[E_j = \begin{cases} 0, & \text{for } j = 1 \text{ and } j = n-r+1, \ldots, n; \\ \leq p_1 + p_2 + \cdots + p_{j-1}, & \text{for } j = 2, \ldots, n-m+1; \\ \leq p_1 + p_2 + \cdots + p_{j-1} - (j-n+m-1)\tau, & \text{for } j = n-m+2, \ldots, n-r; \end{cases} \tag{2}\]

which implies

\[Z^*_c \leq \sum_{j=1}^{n} E_j \leq \sum_{l=1}^{n-r-1} (n-r-l)p_l - \sum_{i=1}^{m-r-1} i\tau, \tag{3}\]

since \(p_l\) appears \(n-r-l\) times in the right-hand side of (2) when we sum up all \(E_1, \ldots, E_n\) and \(\sum_{j=n-m+2}^{n-r} (j-n+m-1)\tau = \sum_{i=1}^{m-r-1} i\tau\). From (1) and (3) and \(n = km\), we have

\[mZ^*_c - Z^*_c \geq \sum_{i=1}^{k-1} (k-i) \sum_{j=1}^{m} p_{(i-1)m+j} - \sum_{i=1}^{km-r-1} (km-r-l)p_l + \sum_{i=1}^{m-r-1} i\tau = \sum_{i=1}^{k-1} \sum_{j=1}^{r} i p_{im-r+j} - \sum_{i=1}^{k} \sum_{j=1}^{m-r-1} i p_{im-r-j} + \sum_{i=1}^{m-r-1} i\tau, \tag{4}\]
where the validity of this equation can easily be verified by using induction on \( k \). Note that, by assumption, \( \sum_{j=1}^{n} p_j \leq m \tau \) and \( p_1 \leq p_2 \leq \cdots \leq p_n \). Hence,

\[
\sum_{i=1}^{k} p_{(i-1)m+1} \leq \tau, \\
\sum_{i=1}^{k} p_{(i-1)m+1} + \sum_{i=1}^{k} p_{(i-1)m+2} \leq 2\tau, \\
\cdots \\
\sum_{i=1}^{k} p_{(i-1)m+1} + \sum_{i=1}^{k} p_{(i-1)m+2} + \cdots + \sum_{i=1}^{k} p_{(i-1)m+(m-r-1)} \leq (m - r - 1)\tau.
\]

Summing up the above \((m - r - 1)\) inequalities, we have

\[
\sum_{j=1}^{m-r-1} (m - r - l) \sum_{i=1}^{k} p_{(i-1)m+l} \leq \sum_{i=1}^{m-r-1} \tau r.
\]

Letting \( j = m - r - l \), we have \( \sum_{j=1}^{m-r-1} (m - r - l) \sum_{i=1}^{k} p_{(i-1)m+l} = \sum_{j=1}^{m-r-1} j \sum_{i=1}^{k} p_{(m-r-j)} \). Thus,

\[
\sum_{i=1}^{k} \sum_{j=1}^{m-r-1} j p_{(m-r-j)} = \sum_{j=1}^{m-r-1} j \sum_{i=1}^{k} p_{(m-r-j)} \leq \sum_{i=1}^{m-r-1} \tau r.
\]

Hence, inequality (4) implies

\[
mZ_c^* - Z_c^{H} \geq 0,
\]
or

\[
Z_c^{H} \leq mZ_c^*.
\]

Notice that the due-date cost of a job ranges from \( \tau \) to \( m\tau \), which implies

\[
Z_d^{H} \leq mZ_d^*.
\]

Therefore,

\[
Z^H = Z_c^{H} + Z_d^{H} \leq m(Z_c^* + Z_d^*) = mZ^*.
\]

For any given constant \( m \), Theorem 2 provides us with a constant worst-case bound of \( m \) on the performance ratio \( Z^H/Z^* \). Note that Step 2 of heuristic H is necessary in order to guarantee the error bound of \( m \). Without Step 2, the performance of the heuristic can be arbitrarily bad. This can be seen from a simple instance with two jobs, each with a processing time 1, and \( \tau = 2 \), \( \beta = 0 \). Without Step 2, the total cost is 1, yet the optimal solution is equal to 0 (provided that \( m \geq 2 \)).

Note that heuristic H obtains a solution to P(m) without considering the due-date cost \( \beta \) at all. In fact, a simple way to improve heuristic H is to add a new step to the algorithm.

**Improved heuristic H' for P(m)**

Step 1: Same as Step 1 of heuristic H.
Step 2: For \( i \leftarrow 1, \ldots, m - 1 \), if no job is currently assigned due-date \( it \), then move the job \( J_{n-i+1} \) to the \( i \)th due-date period and have this job complete at time \( it \), otherwise go to Step 3.

Step 3: Let \( r \) be the number of jobs moved in Step 2. For \( i = 1, \ldots, r \) and \( j = r + 1, \ldots, m \), move the jobs from the \( i \)th due-date period to the \((m - r + i)\)th due-date period and move the jobs from period \( j \) to period \( j - r \).

Note that before the move of jobs in Step 3, each of the due-date periods \( 1, \ldots, r \) has only one job, while none of the periods \( r + 1, \ldots, m \) is empty. Thus, Step 3 of \( H' \) will not increase the total due-date cost, while the total earliness cost will not be affected by this step at all. Therefore, the solution obtained by \( H' \) is no worse than that obtained by \( H \).

5. The equal processing time case

In this section, we consider a special case of problem \( P \) where all the jobs have identical processing times, i.e., \( p_1 = p_2 = \cdots = p_n = p \). In this special case, besides Properties 1, 2, and 3 described in Section 2, we have the following additional properties:

Property 4. There exists an optimal solution such that if a group covers \( k > 1 \) due-date periods, then there are \( \lfloor k \tau / p \rfloor \) jobs in that group, where \( \lfloor x \rfloor \) denotes the largest integer no greater than \( x \).

Proof. Obviously, such a group cannot contain more than \( \lfloor k \tau / p \rfloor \) jobs. Now, suppose that it contains \( q < \lfloor k \tau / p \rfloor \) jobs. Let \( i, i + 1, \ldots, i + k - 1 \) be the due-date periods covered by this group. Then we can move the first \( q - \lfloor (k-1) \tau / p \rfloor \) jobs in this group backward to period \( i \) so that the \((q - \lfloor (k-1) \tau / p \rfloor)\)th job in this group completes at time \( it \). That is, the group is now split into two groups (see Fig. 5). It is easy to see that this splitting of the group will neither increase the total due-date cost nor increase the total earliness penalty. □

Property 5. There exists an optimal solution such that the groups are sequenced in a decreasing order of the number of jobs in the group.

Proof. It can be proved easily by a group interchange argument. □
including earliness and due-date costs) of the group that covers periods \( i \) through \( j \) (\( i < j \)). Let \( u_i = \lfloor \tau/p \rfloor \). When \( i > 1 \), \( u_i \) represents the number of jobs in a group that covers \( i \) due-date periods. Then \( \Gamma_{ij} \) can be evaluated recursively as follows:

\[
\Gamma_{12} = [(u_2 + u_1 + 1)\beta] + \left\lfloor \frac{1}{2}u_2(u_2 - 1) p - (u_2 - u_1 - 1)\tau \right\rfloor,
\]

where the first term on the right-hand side of the equation is the due-date cost and the second term is the earliness cost of the group (see Fig. 6). For \( j = 3, \ldots, n \),

\[
\Gamma_{1j} = \Gamma_{1,j-1} + [(u_j + 1)\beta] + \left\lfloor \frac{1}{2}(u_j(u_j - 1) - u_{j-1}(u_{j-1} - 1)) p - (u_j - u_{j-1} - 1)(j - 1)\tau - (j - 2)\tau \right\rfloor,
\]

where the second term on the right-hand side is the difference in due-date cost between \( \Gamma_{1j} \) and \( \Gamma_{1,j-1} \), and the third term is the difference in earliness cost between these two groups (see Fig. 7). For \( i = 2, \ldots, n-1 \) and \( j = i+1, \ldots, n \),

\[
\Gamma_{ij} = \Gamma_{i-1,j-1} + u_{j-i+1}\beta.
\]

Hence, the values of all \( \Gamma_{ij} \) (\( i < j \)) can be evaluated in \( O(n^2) \) time.

Note that if a group covers only one period, it may have less than \( u_1 \) jobs. We define \( \hat{\Gamma}_j (i) \) as the minimum cost (including earliness and due-date costs) to assign \( i \) jobs to due-date periods \( i, i+1, \ldots \) with at most \( u_1 \) jobs per period. When \( i = 1 \), \( \hat{\Gamma}_1(1), \hat{\Gamma}_1(2), \ldots, \hat{\Gamma}_1(n) \) can be evaluated using the following method. For every due-
each of these
$u_2 - u_1 - 1$ jobs has
due-date cost $\beta$
each of these
$u_1 + 1$ jobs has
due-date cost $2\beta$

(u_2 jobs in this group)

(i) Due-date cost of this group = $(u_2 - u_1 - 1)\beta + (u_1 + 1)2\beta = (u_2 + u_1 + 1)\beta$

total earliness of these $u_2 - u_1 - 1$ jobs
$$= \sum_{i=1}^{u_2 - u_1 - 1} [(i - 1)p + \Delta]$$
$$= \frac{1}{2}u_2(u_2 - 1)p - \frac{1}{2}u_1(u_1 + 1)p - (u_2 - u_1 - 1)p$$

(total earliness of these $u_1 + 1$ jobs
$$= \sum_{i=1}^{u_1 + 1} (i - 1)p - \frac{1}{2}u_1(u_1 + 1)p$$

(ii) Earliness cost of this group
$$= \frac{1}{2}u_2(u_2 - 1)p - (u_2 - u_1 - 1)p$$

Fig. 6. The formula for $\Gamma_{12}$.

date period $i = 1, \ldots, n$, define "position $(i, k)$" $(k = 1, \ldots, u_1)$ as shown in Fig 8. The cost of assigning a job to position $(i, k)$ is given as $h(i, k) = i\beta + (k - 1)p$. We first calculate the values of $h(i, k)$ for $i = 1, \ldots, n$ and $k = 1, \ldots, u_1$. Let $h'(l)$ be the $l$th smallest value among all $h(i, k)$'s. (It is not difficult to see that $h'(1), \ldots, h'(n)$ can be obtained in $O(n^2)$ time.) Then $\hat{f}_1(1)$ is given as

$$\hat{f}_1(1) = h'(1)$$

and for $l = 2, 3, \ldots, n$,

$$\hat{f}_1(l) = \hat{f}_1(l - 1) + h'(l).$$
The total cost of these $u_j-u_{j-1}$ jobs is
\[
= \sum_{i=1}^{u_j-u_{j-1}-1} \left[ (i-1)p + \Delta \right] + (u_j-u_{j-1}-1)\beta
= \frac{1}{2} [u_j(u_j-1) - u_{j-1}(u_{j-1}+1)]p - (u_j-u_{j-1}-1)(j-1)\tau + (u_j-u_{j-1}-1)\beta
\]

earliness and due-date cost of this job
\[
= u_{j-1}p - (j-2)\tau + 2\beta
\]

For $i > 1$, $\hat{\Gamma}_i(l)$ can be evaluated by the formula
\[
\hat{\Gamma}_i(l) = \hat{\Gamma}_{i-1}(l) + l\beta.
\]
Hence, the values of all $\hat{\Gamma}_i(l)$ ($i, l = 1, \ldots, n$) can be evaluated in $O(n^2)$ time.

Given all the values of $\Gamma_{ij}$ and $\hat{\Gamma}_i(l)$, the following dynamic program solves the equal processing time case of problem P.

**Dynamic program for the grouping problem**

1. Define $f(i, j) = \text{minimum cost to assign the subset of jobs } \{J_1, \ldots, J_j\} \text{ into the first } i \text{ due-date periods subject to the constraint that each group must cover at least 2 due-date periods.}$
2. Recurrence relation
\[
f(i, j) = \min_{k=1,\ldots,j-1} \{ f(k-1, j - u_{i-k+1}) + \Gamma_{ki} \}.
\]
(3) Boundary conditions
\[ f(0, j) = \begin{cases} 
0, & \text{if } j = 0; \\
+\infty, & \text{if } j > 0; 
\end{cases} \]
\[ f(1, j) = +\infty, \quad \forall j. \]

(4) Objective
\[ \min_{\substack{0 \leq l \leq \lfloor n/2 \rfloor \\
\hat{l}, \hat{r} \leq \hat{u}}} \{ f(i, j) + \hat{l}_{i+1}(n - j) \}. \]

The validity of this objective follows directly from Property 5. This dynamic program has a computational complexity of
\[ O \left( \sum_{i=0}^{\lfloor n/u_1 \rfloor} (u_i - iu_1 + 1)i \right) \leq O((n/u_1)^3). \]

Thus, the above procedure solves problem P with identical processing times in \( O(n^3) \) time. Note that this running time is only pseudo-polynomial since the input of the identical processing time problem can be encoded using only \( O(\log n + \log \tau + \log p) \) bits. However, this solution procedure is much more efficient than the general dynamic program described in Section 3. Note also that this algorithm can easily be modified to solve problem \( P(m) \) with identical processing times as well.

6. Conclusions

We have analyzed the complexity and heuristics of the Fixed Interval Due-Date Scheduling Problem. We proved that problem P is NP-hard in the strong sense, while problem \( P(m) \) is pseudo-polynomial time solvable if \( m \) is a fixed number. A simple heuristic was introduced to solve problem \( P(m) \) and worst-case analysis was performed. We proved that, when \( m \) is a constant, our heuristic has a worst-case bound of \( Z^H/Z^* \leq m. \) An \( O(n^3) \) time dynamic programming algorithm was also developed to solve the special case where all the job processing times are identical.

Acknowledgements

We would like to thank the referees for their helpful comments and suggestions.

References


