RAIRO-Oper. Res. 55 (2021) 1787–1798
https://doi.org/10.1051/ro/2021079

RAIRO Operations Research
www.rairo-ro.org

MULTITASKING SCHEDULING PROBLEMS WITH A COMMON DUE-WINDOW

Chen Xu[1,*], Yinfeng Xu[1], Feifeng Zheng[1] and Ming Liu[2]

Abstract. We study multitasking scheduling and due-window assignment problems in a single machine, which can be found in various application domains. In multitasking environment, unfinished job always interrupts in-processing job. In common due window assignment, the aim is to find optimal due window to minimise the value of the earliness and tardiness penalty. In this paper, we study two problems, where the objective of the first problem is minimise the earliness, tardiness, due-window starting time, and due-window size costs, the objective of the second problem is minmax common due-date with completion time penalty, then we obtain some analytical properties and provide polynomial time solutions. Finally, the experimental results show that the proposed methods are effective.

Mathematics Subject Classification. 90B35.

Received June 8, 2020. Accepted May 9, 2021.

1. Introduction

Multitasking scheduling has attracted much attention in recently years. The multitasking phenomenon can be frequently observed in our daily lives. One of the examples is human multitasking in which a continuing schedule is interrupted by many routine activities such as sending e-mail or making end-of-day file backup in the workplace today. A possible reason is why people multitask are interrupted by clients or colleagues, another possible reason is due to a personality trait because working the same things for a long time is ineffective [15,16].

At the same time, commom due window assignment has remained an important topic in scheduling research. In business, managers usually need to arrange a common due-window. In such a case, a batch of work piece are ordered by the same customer, because of the different completion time of each work piece, the cost of batch shipment is relatively high. In order to save transportation costs, it is usually uniform shipment after all the work pieces are processed, so managers should make a common due date which is a predetermined time point. But the customers may have some amount of inventory when customers order a batch of work piece. Furthermore, there may be some force majeure factors during the long-distance transportation of goods, most customers can tolerant the common due date. Moreover, Just-in-time scheduling can improve customer satisfaction, but the cost is higher. So we relax the assumption of a common due date to a common due window.

Keywords. Multitasking scheduling, due-window, earliness-tardiness, minmax, assignment problem.

[1] Donghua University, Shanghai, P.R. China.
[2] Tongji University, Shanghai, P.R. China.
*Corresponding author: 1123315955@qq.com

© The authors. Published by EDP Sciences, ROADEF, SMAI 2021

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

To the best of our knowledge, We are the first to study the multitasking scheduling with common due-window. Our contributions mainly include:

(1) We consider the multitasking scheduling with common due-window to minimize two objectives, and give some analytical results.
(2) We propose two polynomial-time algorithms for the two problems.
(3) We do some numerical experiments to demonstrate the efficiency of the algorithm, and give some future issues.

Owing to the practical significance of common due window scheduling in the multitasking environment, there is a need to study the problem under consideration in this research. We organise the rest of the paper as follows: In Section 2, we introduce some literature about multitasking and common due-window. In Section 3, we give some definitions, then we introduce new multitasking model and give analytical results. In Section 4, We propose a polynomial-time algorithm for the two problems, and give some numerical experiments to demonstrate the efficiency of the algorithm. Finally, we conclude the paper and suggest topics for future research in Section 5.

2. Literature interview

Research on multitasking scheduling was in initiated by Hall *et al.* [3], who considered interruption and switching factors, in which a job was interrupted by other unfinished jobs, then they developed some new solution algorithms to solve the classical scheduling model (*i.e.* the total completion time). Hall *et al.* [4] considered two scheduling system, which are alternate period processing and shared processing. Zhu *et al.* [24] considered a rate-modifying activity (RMA) into multitasking scheduling to minimize the maskspan, total completion time, maximum lateness, and due-date assignment related cost. Zhu *et al.* [25] then extended their work to consider multiple rate-modifying activities to minimise the other objectives about the total completion time, the total waiting time, the total absolute differences in completion time and the total absolute differences in completion time. Liu *et al.* [12] considered multitasking scheduling on a single machine with the common due-date assignment to minimize the weighted earliness, tardiness, and due date assignment cost, and developed the DDA algorithm to solve the problem. Ji *et al.* [7] studied parallel-machine scheduling with machine-dependent due-window assignment to minimize the total cost that comprises the earliness, tardiness, and due-window related costs. Xiong *et al.* [19] considered the multitasking scheduling problem on unrelated parallel machines to minimize the total weighted completion time and propose an exact branch-and-price algorithm. Li *et al.* [8] addressed several two-agent scheduling problems in the presence of multitasking, polynomial and pseudo-polynomial time algorithms are proposed to solve the setting, involving various combinations of cost functions. Wang *et al.* [17] considered a multitasking scheduling model with multiple agents and ascertained the computational complexity status of each of the problems. Yang *et al.* [21] addressed a two-agent scheduling problem with due date assignment under multitasking environment, they showed that the problem is NP-hard and devised a pseudo-polynomial time dynamic programming algorithm. Wang *et al.* [18] investigated a multitasking scheduling and due date assignment problem with position-dependent deterioration effect and efficiency promotion, then they designed an efficient polynomial time algorithm for several scheduling criteria including the makespan, the total completion time, and two due date-related criteria.

For common due window assignment problems, Liman *et al.* [9] considered a single machine static and deterministic scheduling problem, the objective is to find the optimal size and location the window and an optimal sequence to minimise earliness, tardiness, window size and window location, they proposed an $O(n \log n)$ algorithm to solve the problem. Gur mosheiov *et al.* [13] addressed a common due-window assignment problem on parallel identical machines. Adam Janiak *et al.* [5] studied problems of scheduling n unit-time jobs on m identical parallel machines to minimize a weighted sum or maximum of costs associated with job earliness, job tardiness and due window location and size, establish properties of optimal solutions of these min-sum and min-max problems and reduce them to min-sum or min-max assignment problems solvable in $O(n^5/m^2)$ and $O(n^{4.5} \log^{0.5} n/m^2)$ time. Yin *et al.* [22] addressed a batch delivery single-machine scheduling problem in which

jobs have an assignable common due window and showed proposed problem can be optimally solved in $O(n^8)$ time by a dynamic programming algorithm. Yin *et al.* [22] consider single-machine batch delivery scheduling with an assignable common due date and controllable processing times and provided an $O(n^5)$ dynamic programming algorithm to find the optimal job sequence. Enrique Gerstl *et al.* [2] studied both cases of a non-restrictive and a restrictive due-window, for both settings they introduce algorithms requiring $O(2^m n^3)$ time and $O(3^m n^3)$ time, respectively, where n is the number of jobs and m is the number of uniform machines. Recently, due window assignment problems have been studied, among others, by Yang *et al.* [20], Ji *et al.* [6], Liu *et al.* [10], Liu *et al.* [12], etc.

3. Problem statement

We assume that n jobs $J = \{J_1, J_2, \ldots, J_n\}$ are processed on a single machine, and the start time is zero, each job J_j has a normal processing time p_j, and the machine is working all the time. There is no idle time between any two adjacent jobs. All jobs have common due window, the due-window starting time is d_1, the due-window finishing time is d_2, D is equal to $d_2 - d_1$, which is defined the window size. Jobs completed in due-window incurs no penalties, other jobs incurs either earliness or tardiness penalties. If the job is early job, then $E_j = \max\{0, d_1 - C_j\}$, if the job is tardy job, then $T_j = \max\{0, C_j - d_2\}$. Only one job can be processed at a certain time. This job is called the primary job, the interrupted and unfinished jobs are called the waiting jobs of the primary job. Every waiting job must interrupt the primary job one time, which we called as the interruption time. The interruption time caused by job i during the processing of j is given by function $g_i(p_i)$, where $0 \leq g_i(p_i) < p_i$. The above studies all assume that the interruption function $g_k(p'_k)$ is Dp_k, where $0 < D < 1$, in fact, the interrupted time of waiting jobs is uncertain. But for the convenience of calculation, it is also considered as a function of the remaining time of waiting job in this paper. The time during finished time of the former job and start time of the next job is referred to as the switching time, which function is defined as $f(|S_j|)$. We assume that the switching time during every two job is same, if the number of the waiting jobs of primary job is k, the $f(|S_j|)$ is equal to $k\delta$. In other words, the remaining processing time of the waiting job will decrease because a part of the processing time is completed in the interruptions of previous jobs. When job j is the primary job, we denote S_j as all waiting jobs. Each job can be arranged as primary job a time, the multitasking function is defined as $f(|S_j|) + \sum_{i \in S_j} g_i(p_i)$, which expresses the total amount of interruption during the processing of job j.

As in Hall *et al.* [3], we define the remaining processing time p_i of a waiting job i as a function $h_i(\cdot) := \{1, 2, \ldots, n\} \to R$. Specifically, with respect to interruption function $g_i(p_i)$, the remaining processing time of job i after it has interrupted l primary jobs is given by $h_i(l)$, where $1 \leq l \leq n$. That is, $h_i(l)$ is the remaining processing time of job i when it is considered at the lth position in a schedule.

$$
\begin{aligned}
&h_i(1) = p_i;\\
&h_i(2) = h_i(1) - g_i(h_i(1));\\
&h_i(3) = h_i(2) - g_i(h_i(2));\\
&\cdots\\
&h_i(l) = h_i(l-1) - g_i(h_i(l-1));\\
&\cdots\\
&h_i(n) = h_i(n-1) - g_i(h_i(n-1));
\end{aligned}
$$

Observation 1. $C_{\max}$ is a constant in an optimal schedule.

Proof. For any scheduling, the total completion time is divided into three parts. The first part is the working time of the main job, the second part is the working time of the waiting job, and the third part is the switching time between the waiting jobs. The main job is a interrupted job, its working time is low p_i, but the rest of p_i is accomplished on front of itself. So the working time is always equal to p_i. What's more, the number of

jobs switching is always same for any schedule, in this paper, we consider the switching function is a index 1 function of the number of jobs, so the switching time is equal. The completion time of the last job is equal to the processing time and the switching time of all jobs, so it is same. □

4. Multitasking scheduling problem

We study two problems, the first objective is minimise $\sum(\alpha E_j + \beta T_j + \gamma d_1 + \delta D)$, every early job has $(\alpha E_j + \gamma d_1 + \delta D)$ cost, every late job has $(\alpha T_j + \gamma d_1 + \delta D)$ cost, in order to minimise $\sum(\alpha E_j + \beta T_j + \gamma d_1 + \delta D)$ cost, we hope to minimise the highest cost among all jobs, so another is to minimized $\max_{1\leq j\leq n}\{\max\{\alpha E_j + \gamma d_1 + \delta D, \beta T_j + \gamma d_1 + \delta D\}\}$, which α is the unit penalty for earliness, β is the unit penalty for tardiness, γ is the unit penalty for the due-window starting time, δ is the unit penalty for the window size.

4.1. Common due-window problem

A previous study on common due-window assignment problem is given by Gur Mosheiov, and Assaf Sarig. They consider the problem $1 \parallel \sum(\alpha E_j+\beta T_j+\gamma d_1+\delta D)$. We study the problem $1 \mid mt \mid \sum(\alpha E_j+\beta T_j+\gamma d_1+\delta D)$, if the interruption function $g(\cdot) \equiv 0$ and the switching function $f(\cdot) \equiv 0$ for all jobs in multitasking environment, which is reduced to $1 \parallel \sum(\alpha E_j + \beta T_j + \gamma d_1 + \delta D)$. So we have some similar results in this case.

Lemma 4.1. *For the problem* $1 \mid mt \mid \sum(\alpha E_j + \beta T_j + \gamma d_1 + \delta D)$*, if* $\gamma > \delta$*, an optimal schedule exists in which the due window starts at time zero.*

Proof. Given any schedule σ and $d_1 \geq 0$, we shift d_1, $\triangle$ units of time to the left, the value of the first term $\sum \alpha E_j$ decreases $\alpha k\triangle$, where k denotes the numbers of early jobs, the value of the second term $\sum \beta T_j$ keeps the same with unchanged, the value of the third term $\sum \gamma d_1$ decreases $\gamma n\triangle$, while the value of the fourth term $\sum \delta D$ increases $\delta n\triangle$. Then the change in the total cost is given by: $\triangle Z = (\delta - \gamma)n\triangle$ - $\alpha k\triangle \leq 0$. Therefore, implying the optimal $d_1 = 0$.

Given the optional $d_1 = 0$, we observe that the problem $1 \mid mt \mid \sum(\alpha E_j + \beta T_j + \gamma d_1 + \delta D)$ reduces to the problem the $1 \mid mt \mid \sum(\beta T_j + \delta d_2)$. The problem $1 \mid mt \mid \sum(\beta T_j + \delta d_2)$ is studied where $\delta > \beta$, the SPT rule is optimal for scheduling jobs and the optimal $d_2 = 0$. □

Proposition 4.2. *For the problem* $1 \mid mt \mid \sum(\alpha E_j + \beta T_j + \gamma d_1 + \delta D)$*, if* $\beta < \delta < \gamma$*, an optimal schedule exists in which the due-window is reduced to a common due-date that starts at time zero.*

Proof. Since $\gamma > \delta$, an optimal schedule exsits the d_1 is equal to 0, then $\delta > \beta$, the optimal $d_2 = 0$. If $\beta < \delta < \gamma$, we conclude that the optimal $d_1 = d_2 = 0$. □

Proposition 4.3. *For the problem* $1 \mid mt \mid \sum(\alpha E_j + \beta T_j + \gamma d_1 + \delta D)$*, if* $\beta < \min\{\delta, \gamma\}$*, an optimal schedule exists in which the due-window is reduced to a common due-date that starts at time zero.*

Proof. (i) $\beta < \gamma < \delta$, suppose that the $d_1 > 0$, we shift d_2, $\triangle$ units of time to the left, the value of the first term $\sum \alpha E_j$ is unchanged, the value of the second term $\sum \beta T_j$ increases $\beta k\triangle$, where k denotes the numbers of tardy jobs, the value of the third term $\sum \gamma d_1$ is unchanged, while the value of the fourth term $\sum \delta D$ decreases $\delta n\triangle$. Then the change in the total cost is given by: $\triangle Z = (\beta k - \delta n)\triangle < (\beta k - \delta k)\triangle = (\beta - \delta)k\triangle < 0$. A further shift of d_2 to coincide with d_1 can reduce the cost. The problem $1 \mid mt \mid \sum(\alpha E_j+\beta T_j+\gamma d_1+\delta D)$ reduces to the problem the $1 \mid mt \mid \sum(\alpha E_j + \beta T_j + \delta d_1)$.

(ii) $\beta < \delta < \gamma$, proposition 1 proposed that the optimal $d_1 = d_2 = 0$.

So if $\beta < \min\{\delta, \gamma\}$, an optimal schedule exists in which the due-window is reduced to a common due-date that starts at time zero. □

Lemma 4.4. *For the problem* $1 \mid mt \mid \sum(\alpha E_j + \beta T_j + \gamma d_1 + \delta D)$*, an optimal schedule exists in which the due-window starting time and the due-window completion time coincide with the completion time of a job in a schedule.*

Proof. We suppose $C_{[k]} < d_1 < C_{[k+1]}$, $C_{[h]} < d_2 < C_{[h+1]}$, for some $k, h \in [1, n]$, we first discuss the d_1, the following two cases are discussed:

(i) We denote $\triangle_1 = C_{[k+1]} - d_1$ increasing the d_1 by $\triangle_1$, the value of the first term $\sum \alpha E_j$ increases by $\alpha k \triangle_1$, the value of the second term $\sum \beta T_j$ is unchanged, the value of the third term $\sum \gamma d_1$ increases by $\gamma n \triangle_1$, the value of the third term $\sum \delta D$ decreases by $\gamma n \triangle_1$, then the total increase of the objective function value is $[\alpha k + (\gamma - \delta)n]\triangle_1$.

(ii) We denote $\triangle_2 = d_1 - C_{[k]}$ decreasing the d_1 by $\triangle_2$,the value of the first term $\sum \alpha E_j$ decreases by $\alpha k \triangle_2$, the value of the second term $\sum \beta T_j$ is unchanged, the value of the third term $\sum \gamma d_1$ decreases by $\gamma n \triangle_2$, the value of the third term $\sum \delta D$ increases by $\gamma n \triangle_2$, then the total decrease of the objective function value is $[\alpha k + (\gamma - \delta)n]\triangle_2$.

Notice that in both case, the objective function can express that $G\triangle, \triangle \in [C_{[k]} - d_1, C_{[k+1]} - d_1], G = [\alpha k + (\gamma - \delta)n]$, is a constant, if $G > 0, \triangle = C_{[k]} - d_1$, it means $d_1 = C_{[k]}$, if $G < 0, \triangle = C_{[k+1]} - d_1$, it means $d_1 = C_{[k+1]}$, while $G = 0$, $d_1 = C_{[k]}$ or $C_{[k+1]}$.

The proof about d_2 is similar to above about d_1.

Therefore, an optimal schedule exists such that both d_1 and d_2 coincide with job completion times. □

Lemma 4.5. *For the problem* $1 \mid mt \mid \sum(\alpha E_j + \beta T_j + \gamma d_1 + \delta D)$*, an optimal schedule exists in which* $d_1 = C_{[k]}$*,* $d_2 = C_{[h]}$*, where* $k = \lceil n(\delta - \gamma)/\alpha \rceil$ *and* $h = \lceil n(\beta - \delta)/\beta \rceil$*.*

Proof. By the proof of Lemma 4.4, we define a function $f(k) = [\alpha k + (\gamma - \delta)n]\triangle_2$, where $k \in \{1, 2, \ldots, n\}$, the function $f(\cdot)$ is increasing in k because $\alpha > 0$, moreover, $f(1) = [\alpha + (\gamma - \delta)n]\Delta 2 < 0$, since $\gamma < \delta$, $(\gamma - \delta)n$ is significantly less than α. To find an optimal d_1, it suffices to find a $k \in \{1, 2, \ldots, n\}$ such that $f(k) \le 0$ and $f(k+1) > 0$. If $f(k) \le 0$ then $k \le n(\delta - \gamma)/\alpha + 1$, as k is an integer, so $k = \lceil n(\delta - \gamma)/\alpha \rceil$.

The proof about d_2 is similar to above about d_1.

Given a sequence $\sigma = [J_{[1]}, J_{[2]}, \ldots, J_{[n]}]$ and $d_1 = C_{[k]}$, $d_2 = C_{[h]}$, where $k = \lceil n(\delta - \gamma)/\alpha \rceil$, $h = \lceil n(\beta - \delta)/\beta \rceil$, the first k jobs are completed early or on time, while the last $(n - h)$ jobs are delayed. So we can reformulate the objective functions as follows.

$$
\begin{aligned}
Z &= \alpha \sum_{j=1}^{n} E_j + \beta \sum_{j=1}^{n} T_j + \gamma n d_1 + \delta n D \\
&= \alpha \sum_{j=1}^{k} (d_1 - C_{[j]}) + \beta \sum_{j=h+1}^{n} (C_{[j]} - d_2) + \gamma n \sum_{j=1}^{k} P^A_{[j]} + \delta n \sum_{j=k}^{h} P^A_{[j]} \\
&= \alpha \sum_{j=1}^{k} (j-1) P^A_{[j]} + \beta \sum_{j=h+1}^{n} (n - j + 1) P^A_{[j]} + \gamma n \sum_{j=1}^{k} P^A_{[j]} + \delta n \sum_{j=k}^{h} P^A_{[j]} \\
&= \sum_{j=1}^{n} \varphi_j P^A_{[j]} \\
&= \sum_{j=1}^{n} \varphi_j [(1-D)^{j-1} P_{[j]} + \omega(n-j) + D(1-D)^{j-1} \sum_{l=j+1}^{n} P_{[l]}] \\
&= \sum_{j=1}^{n} \varphi_j [(1-D)^{j-1} P_{[j]}] + \sum_{j=1}^{n} \varphi_j D(1-D)^{j-1} \sum_{l=j+1}^{n} P_{[l]} + \sum_{j=1}^{n} \varphi_j \omega (n-j) \\
&= \sum_{j=1}^{n} \varphi_j [(1-D)^{j-1} P_{[j]}] + \sum_{j=1}^{n} \sum_{l=1}^{j-1} D(1-D)^{l-1} \varphi_l P_{[j]} + \sum_{j=1}^{n} \varphi_j \omega (n-j)
\end{aligned}
$$

$$= \sum_{j=1}^{n}[\varphi_j(1-D)^{j-1} + \sum_{l=1}^{j-1} D(1-D)^{l-1}\varphi_l]P_{[j]} + \sum_{j=1}^{n}\varphi_j\omega(n-j)$$
$$= \sum_{j=1}^{n}\psi_j P_{[j]} + \sum_{j=1}^{n}\phi_j$$

where $\psi_j = [\varphi_j(1-D)^{j-1} + \sum_{l=1}^{j-1} D(1-D)^{l-1}\varphi_l]$,$\phi_j = \varphi_j\omega(n-j)$

$$\varphi_j = \begin{cases} \alpha(j-1)+\gamma n, & j = 1,\ldots,k; \\ \delta n, & j = k+1,\ldots,h \\ \beta(n-j+1), & j = h+1,\ldots,n. \end{cases}$$

Hence, the problem can be formulated as following assignment problem:

$$\min \sum_{j=1}^{n}\sum_{r=1}^{n} B_{\mathrm{jr}}x_{\mathrm{jr}}$$
$$\sum_{r=1}^{n} x_{\mathrm{jr}} = 1, j = 1,2,\ldots,n,$$
$$\sum_{j=1}^{n} x_{\mathrm{jr}} = 1, r = 1,2,\ldots,n,$$
$$x_{\mathrm{jr}} = 0 \text{ or } 1,\ j,r = 1,2,\ldots,n,$$

where $B_{\mathrm{jr}} = [\varphi_r(1-D)^{r-1} + \sum_{l=1}^{r-1} D(1-D)^{l-1}\varphi_l]p_j$.

Based on the above analysis, we propose the following optimization algorithm to solve the $1 \mid mt \mid \sum(\alpha E_j + \beta T_j + \gamma d_1 + \delta D)$ problem. □

Algorithm 1.

Step 1. Compute d_1 and d_2 at the competition time of the kth job and hth job, where $k = \lceil n(\delta-\gamma)/\alpha \rceil$, $h = \lceil n(\beta-\delta)/\beta \rceil$
Step 2. For i from 1 to n do
  Step 2.1. $h_i(1) = p_i$.
  Step 2.2. For l from 1 to $n-1$ do
    $h_i(l+1) := h_i(l) - g_i(h_i(l))$
Step 3. Compute all B_{jr} for $j,r = 1,\ldots,n$
Step 4. Solve the assignment to determine the local optimal and the total cost
Step 5. Schedule the jobs with multitasking in order of σ

Property 4.6. For the problem $1 \mid mt \mid \sum(\alpha E_j + \beta T_j + \gamma d_1 + \delta D)$, an optimal schedule σ can be obtained in $O(n^3)$ time.

Proof. Step 1 of algorithm 1 can be solved with constant time. Step 2 of algorithm 1, as the processing step by the remaining processing times of all jobs after they have interrupted j primary jobs, for $j = 1,\ldots,n$, are computed, required $O(n^2)$ times, step 4 of algorithm 1 can be obtained in $O(n^2)$ times. The classic assignment problem can be solved with $O(n^3)$ Papadimitriou and Steiglitz [14] and Brucker [1]. Thus the above theorem holds. □

TABLE 1. Job-position processing times $h_i(l)$.

Jobs	Position							
	1	2	3	4	5	6	7	8
1	15	13.50	12.15	10.94	5.47	4.92	4.43	3.99
2	9	8.10	7.29	6.56	5.90	5.31	4.78	4.30
3	26	23.40	21.06	18.95	17.06	15.35	13.81	12.44
4	104	93.60	84.24	75.82	68.23	61.41	55.27	49.74
5	10	9	8.1	7.29	5.90	5.31	4.78	4.30
6	2	1.8	1.62	1.46	1.31	1.18	1.06	0.96
7	25	22.5	20.25	18.23	16.40	14.76	13.29	11.96
8	82	73.8	66.42	59.78	53.80	48.42	43.58	39.22

TABLE 2. Job-position processing times B_{jr}.

Jobs	Position							
	1	2	3	4	5	6	7	8
1	1800	1644	1500	1357	589	457	353	270
2	1080	987	900	813	636	494	381	291
3	3120	2850	2599	2350	1838	1427	1100	843
4	12480	11400	10397	9402	7350	5709	4404	3369
5	1200	1096	1000	904	636	494	381	291
6	240	219	200	181	141	110	84	65
7	3000	2741	2499	2261	1767	1372	1059	810
8	9840	8989	8198	7413	5796	4502	3472	2656

Example 4.7. There are $n = 8$ jobs, the normal processing time are $p_1 = 15, p_2 = 9, p_3 = 26, p_4 = 104, p_5 = 10, p_6 = 2, p_7 = 25, p_8 = 82$,$\omega = 0.1$ and $D = 0.1$. The cost parameters are: $\alpha = 2, \beta = 25, \gamma = 15, \delta = 15.6$.

Then, we compute:

$$k = \lceil 8 \times (15.6 - 15)/2 \rceil = 3, h = \lceil 8 \times (25 - 15.6)/25 \rceil = 4$$
$$\alpha(j-1) + \gamma n = (120, 122, 124).$$
$$\delta n = (124.8)$$
$$\beta(n - j + 1) = (100, 75, 50, 25)$$
$$\varphi_r = (120, 122, 124, 124.8, 100, 75, 50, 25)$$
$$\psi_r = (120, 121.8, 123.42, 124, 107.73, 92.97, 79.68, 67.73).$$

We solve the 8×8 job-position processing times problem given in Table 1, the results are shown in Table 2.

As specified above, the job sequence is (6,2,5,7,1,3,8,4), the completion of the first four jobs are $(C_1, C_2, C_3, C_4) = (29.8,\ 64.79,\ 98.87,\ 141.16)$. The optimal $d_1 = 98.87$, $d_2 = 141.16$, the due-window is of size 42.29, three jobs are early, one job is scheduled inside the window, and four jobs are tardy. The total cost is $Z = 13664.12$.

4.2. Minmax common due-window problem

In this section, the aim is to find the optimal sequence and the start time and size of the due-window that minimize the maximum cost. To do this, P2 is formulated as a linear programming (LP) problem.

$$\begin{aligned}
&\min Z\\
&\quad Z \geq \alpha E_j + \gamma d_1 + \delta D, j = 1, 2, \ldots, n,\\
&\quad Z \geq \beta T_j + \gamma d_1 + \delta D, j = 1, 2, \ldots, n,\\
&\quad d_1, D, Z \geq 0, j = 1, 2, \ldots, n,
\end{aligned}$$

Property 4.8. There are four different cases with specific cost functions:

Case 1. $\beta < \gamma, \delta \geq \beta$, the due window is reduced to a due date at time zero, implying that all jobs are tardy.

$$d_1 = d_2 = 0, Z = \beta C_{\max}$$

Case 2. $\beta \geq \gamma, \delta > \gamma$ and $\delta < \beta\frac{\alpha+\gamma}{\alpha+\beta}$, the due window is again reduced to a due date in the interval $(0, C_{\max})$, implying that there are early and tardy jobs.

$$d_1 = d_2 = \frac{\alpha C_{\min} + \beta C_{\max}}{\alpha + \beta}, Z = \frac{\alpha(\gamma - \beta)C_{\min} + \beta(\alpha + \gamma)C_{\max}}{\alpha + \beta}$$

Case 3. $\beta \geq \gamma, \delta > \gamma$ and $\delta \geq \beta\frac{\alpha+\gamma}{\alpha+\beta}$, the start time of the due window coincides with the completion time of the first job and all jobs are on time.

$$d_1 = C_{\min}, d_2 = C_{\max}, Z = \gamma C_{\min} + \delta(C_{\max} - C_{\min})$$

Case 4. ($\beta \geq \gamma$ and $\delta \leq \gamma$) or ($\beta < \gamma$ and $\delta < \beta$), the due window achieves its maximal size and all jobs are on time.

$$d_1 = 0, d_2 = C_{\max}, Z = \delta C_{\max}$$

So we can reformulate the objective functions as follows.

$$\begin{aligned}
Z &= \sum_{j=1}^{n} W_j P^A_{[j]}\\
&= \sum_{j=1}^{n} W_j[(1-D)^{j-1}P_{[j]} + \omega(n-j) + D(1-D)^{j-1}\sum_{l=j+1}^{n} P_{[l]}]\\
&= \sum_{j=1}^{n} W_j[(1-D)^{j-1}P_{[j]}] + \sum_{j=1}^{n} W_j D(1-D)^{j-1}\sum_{l=j+1}^{n} P_{[l]} + \sum_{j=1}^{n} W_j\omega(n-j)\\
&= \sum_{j=1}^{n} W_j[(1-D)^{j-1}P_{[j]}] + \sum_{j=1}^{n}\sum_{l=1}^{j-1} D(1-D)^{l-1}W_l P_{[j]} + \sum_{j=1}^{n} W_j\omega(n-j)\\
&= \sum_{j=1}^{n}[W_j(1-D)^{j-1} + \sum_{l=1}^{j-1} D(1-D)^{l-1}W_l]P_{[j]} + \sum_{j=1}^{n} W_j\omega(n-j)\\
&= \sum_{j=1}^{n} \psi_j P_{[j]} + \sum_{j=1}^{n} \phi_j
\end{aligned}$$

where $\psi_j = [W_j(1-D)^{j-1} + \sum_{l=1}^{j-1} D(1-D)^{l-1}W_l]$,$\phi_j = W_j\omega(n-j)$

$$W_j = \begin{cases} \beta & \beta < \gamma, \delta \geq \beta \\ \begin{cases} \gamma & j = 1 \\ \frac{\beta(\alpha+\gamma)}{\alpha+\beta} & j = 2, \ldots, n \end{cases} & \beta \geq \gamma, \delta > \gamma \text{ and } \delta \geq \beta\frac{\alpha+\gamma}{\alpha+\beta} \\ \begin{cases} \gamma & j = 1 \\ \delta & j = 2, \ldots, n \end{cases} & \beta \geq \gamma, \delta > \gamma \text{ and } \delta < \beta\frac{\alpha+\gamma}{\alpha+\beta} \\ \delta & (\beta \geq \gamma \text{ and } \delta \leq \gamma) \text{ or } (\beta < \gamma \text{ and } \delta < \beta). \end{cases}$$

Hence, the problem can be formulated as following assignment problem:

$$\begin{aligned} \min & \sum_{j=1}^{n}\sum_{r=1}^{n} C_{\mathrm{jr}}x_{\mathrm{jr}} \\ & \sum_{r=1}^{n} x_{\mathrm{jr}} = 1, j = 1, 2, \ldots, n, \\ & \sum_{j=1}^{n} x_{\mathrm{jr}} = 1, r = 1, 2, \ldots, n, \\ & x_{\mathrm{jr}} = 0 or 1 j, r = 1, 2, \ldots, n, \end{aligned}$$

where $C_{\mathrm{jr}} = [\varphi_r(1-D)^{r-1} + \sum_{l=1}^{r-1} D(1-D)^{l-1}\varphi_l]p_j$.
The due date are given by $d_1 = \sum_{j=1}^{n} W_j^{d_1} P_{[j]}^A$, $d_2 = \sum_{j=1}^{n} W_j^{d_2} P_{[j]}^A$. Where

$$W_j^{d_1} = \begin{cases} 0 & \beta < \gamma, \delta \geq \beta \\ \begin{cases} 1 & j = 1 \\ \frac{\beta}{\alpha+\beta} & j = 2, \ldots, n \end{cases} & \beta \geq \gamma, \delta > \gamma \text{ and } \delta \geq \beta\frac{\alpha+\gamma}{\alpha+\beta} \\ \begin{cases} 1 & j = 1 \\ 0 & j = 2, \ldots, n \end{cases} & \beta \geq \gamma, \delta > \gamma \text{ and } \delta < \beta\frac{\alpha+\gamma}{\alpha+\beta} \\ 0 & (\beta \geq \gamma \text{ and } \delta \leq \gamma) \text{ or } (\beta < \gamma \text{ and } \delta < \beta) \end{cases}$$

$$W_j^{d_2} = \begin{cases} 0 & \beta < \gamma, \delta \geq \beta \\ \begin{cases} 1 & j = 1 \\ \frac{\beta}{\alpha+\beta} & j = 2, \ldots, n \end{cases} & \beta \geq \gamma, \delta > \gamma \text{ and } \delta \geq \beta\frac{\alpha+\gamma}{\alpha+\beta} \\ \begin{cases} 1 & j = 1 \\ 1 & j = 2, \ldots, n \end{cases} & \beta \geq \gamma, \delta > \gamma \text{ and } \delta < \beta\frac{\alpha+\gamma}{\alpha+\beta} \\ 1 & (\beta \geq \gamma \text{ and } \delta \leq \gamma) \text{ or } (\beta < \gamma \text{ and } \delta < \beta). \end{cases}$$

Based on the above analysis, we propose the following optimization algorithm to solve the $1 \mid mt \mid \max\{\max\{\alpha E_j + \gamma d_1 + \delta D, \beta T_j + \gamma d_1 + \delta D\}\}$ problem.

TABLE 3. Case 1: Job-position processing times C_{jr} for $\alpha = 1, \beta = 9, \gamma = 16, \delta = 19$.

Jobs	Position							
	1	2	3	4	5	6	7	8
1	135.00	121.50	109.35	98.46	49.23	44.28	39.87	35.91
2	81.00	72.90	65.61	59.04	53.10	47.79	43.02	38.70
3	234.00	210.60	189.54	170.55	153.54	138.15	124.29	111.96
4	936.00	842.40	758.16	682.38	614.07	552.69	497.43	447.66
5	90.00	81.00	72.90	65.61	53.10	47.79	43.02	38.70
6	18.00	16.20	14.58	13.14	11.79	10.62	9.54	8.64
7	225.00	202.50	182.25	164.07	147.60	132.84	119.61	107.64
8	738.00	664.20	597.78	538.02	484.20	435.78	392.22	352.98

Algorithm 2.

Step 1. Compute d_1 and d_2 at the competition time of the kth job and hth job, where $k = \lceil n(\delta - \gamma)/\alpha \rceil$, $h = \lceil n(\beta - \delta)/\beta \rceil$
Step 2. For i from 1 to n do
  Step 2.1. $h_i(1) = p_i$.
  Step 2.2. For l from 1 to $n - 1$ do
    $h_i(l + 1) := h_i(l) - g_i(h_i(l))$
Step 3. Compute all C_{jr} by for $j, r = 1, \ldots, n$
Step 4. Solve the assignment to determine the local optimal and the total cost
Step 5. Schedule the jobs with multitasking in order of σ

Property 4.9. For the problem $1 \mid mt \mid \max\{\max\{\alpha E_j + \gamma d_1 + \delta D, \beta T_j + \gamma d_1 + \delta D\}\}$, an optimal schedule σ can be obtained in $O(n^3)$ time.

Proof. Step 1 of algorithm 2 can be solved with constant time. Step 2 of algorithm 2, as the processing step by the remaining processing times of all jobs after they have interrupted j primary jobs, for $j = 1, \ldots, n$, are computed, required $O(n^2)$ times, step 3 of algorithm 2 can be obtained in $O(n^2)$ times. The classic assignment problem can be solved with $O(n^3)$ times. Thus the above theorem holds. □

Example 4.10. There are $n = 8$ jobs, the normal processing time are $p_1 = 14, p_2 = 35, p_3 = 28, p_4 = 38, p_5 = 26, p_6 = 49, p_7 = 43$, and $p_8 = 40$.

We solve the 8×8 job-position processing times problem given in Table 1 with four different combinations of the cost parameters, depicting one of the four sub-cases discussed in Property 4.8. The results are shown in Tables 3–7.

5. CONCLUSION

We study the method of common due-window assignment and multitasking scheduling problem. We have proposed two polynomial algorithms to solve the single-machine common due window problem to minimise a cost function based on earliness, tardiness, window size, and due-window starting times, we illustrate the algorithm with a numerical example. We then show that the solutions can be extended to the problem with minmax common due-window assignment.

Future research directions on multitasking issues may include deterioration and positional effects, job rejection, parallel machine.

TABLE 4. Case 2: Job-position processing times C_{jr} for $\alpha = 4, \beta = 15, \gamma = 9, \delta = 14$.

Jobs	Position							
	1	2	3	4	5	6	7	8
1	135.00	136.76	123.08	110.82	55.41	49.84	44.88	40.42
2	81.00	82.05	73.85	66.45	59.77	53.79	48.42	43.56
3	234.00	237.04	213.34	191.96	172.82	155.50	139.90	126.02
4	936.00	948.17	853.35	768.06	691.17	622.08	559.88	503.87
5	90.00	91.17	82.05	73.85	59.77	53.79	48.42	43.56
6	18.00	18.23	16.41	14.79	13.27	11.95	10.74	9.72
7	225.00	227.93	205.13	184.67	166.13	149.52	134.63	121.15
8	738.00	747.59	672.83	605.57	544.99	490.49	441.47	397.30

TABLE 5. Case 3: Job-position processing times C_{jr} for $\alpha = 6, \beta = 19, \gamma = 14, \delta = 15$.

Jobs	Position							
	1	2	3	4	5	6	7	8
1	210	201.2	181	163	81.5	73.3	66.0	59.5
2	126	120.7	108.6	97.7	87.9	79.1	71.2	64.1
3	364	348.7	313.8	282.4	254.2	228.7	205.8	185.4
4	1456	1394.6	1255.2	1129.7	1016.6	915.0	823.5	741.1
5	140	134.1	120.7	108.6	87.9	79.1	71.2	64.1
6	28	26.8	24.1	21.8	19.5	17.6	15.8	14.3
7	350	335.2	301.7	271.6	244.4	219.9	198.0	178.2
8	1148	1099.6	989.7	890.7	801.6	721.5	649.3	584.4

TABLE 6. Case 4: Job-position processing times C_{jr} for $\alpha = 20, \beta = 5, \gamma = 15, \delta = 3$.

Jobs	Position							
	1	2	3	4	5	6	7	8
1	45	40.50	36.45	32.82	16.41	14.76	13.29	11.97
2	27.00	24.30	21.87	19.68	17.70	15.93	14.34	12.90
3	78.00	70.20	63.18	56.85	51.18	46.05	41.43	37.32
4	312.00	280.80	252.72	227.46	204.69	184.23	165.81	149.22
5	30.00	27.00	24.30	21.87	17.70	15.93	14.34	12.90
6	6.00	5.40	4.86	4.38	3.93	3.54	3.18	2.88
7	75.00	67.50	60.75	54.69	49.20	44.28	39.87	35.88
8	246.00	221.40	199.26	179.34	161.40	145.26	130.74	117.66

TABLE 7. Optimal sequence, due-window and cost function for different cases.

Case	$(\alpha, \beta, \gamma, \delta)$	Optimal sequence	$C_{\min}$	$C_{\max}$	d_1	d_2	Z
1	(1,9,16,19)	$(J_6, J_2, J_5, J_7, J_1, J_3, J_8, J_4)$	29.8	275.8	0	0	1355.4
2	(4,15,9,14)	$(J_2, J_6, J_5, J_7, J_1, J_3, J_8, J_4)$	36.1	275.8	225.34	225.34	2241.6
3	(6,19,14,15)	$(J_6, J_2, J_5, J_7, J_1, J_3, J_8, J_4)$	29.8	275.8	29.8	275.8	1522.2
4	(20,5,15,3)	$(J_6, J_2, J_5, J_7, J_1, J_3, J_8, J_4)$	29.8	275.8	0	275.8	451.8

Acknowledgements. This paper has benefited from financial support from the National Science Foundation of China (71771048, 71832001) and the Fundamental Research Funds for the Central Universities.

References

[1] P. Brucker, Scheduling Algorithms. Berlin, Springer (2007).
[2] E. Gerstl and G. Mosheiov, Due-window assignment with identical jobs on parallel uniform machines. *Eur. J. Oper. Res.* **229** (2013) 41–47.
[3] N.G. Hall, J.Y.T. Leung and C.L. Li, The effects of multitasking on operations scheduling. *Prod. Oper. Manag.* **24** (2015) 1248–1265.
[4] N.G. Hall, J.Y.T. Leung and C.L. Li, Multitasking via alternate and shared processing: algorithms and complexity. *Discret. Appl. Math.* **208** (2016) 41–58.
[5] A. Janiak, W. Janiak, M.Y. Kovalyov, E. Kozan and E. Pesch, Parallel machine scheduling and common due window assignment with job independent earliness and tardiness costs. *Inf. Sci.* **224** (2013) 109–117.
[6] M. Ji, X. Zhang, X.Y. Tang, T.C.E. Cheng, G.Y. Wei and Y.Y. Tan, Group scheduling with group-dependent multiple due windows assignment. *Int. J. Pro. Res.* **54** (2016) 1244–1256.
[7] M. Ji, L.J. Liao, W.Y. Zhang, T.C.E. Cheng and Y.Y. Tan, Multitasking Scheduling with General Aging Effect. *Multiple Rate-Modifying Activities and Past-sequence-dependent Delivery Times*. Working Paper (2018).
[8] S. Li, R. Chen, and J. Tian, Multitasking Scheduling Problems with Two Competitive Agents. *Eng. Optimiz.* **52** (2019) 1940–1956.
[9] S.D. Liman, S.S. Panwalkar and S. Thongmee, Common due window size and location determination in a single machine scheduling problem. *J. Oper. Res. Soc.* **93** (1998) 68–74.
[10] L. Liu, J.J. Wang and X.Y. Wang, Single machine due-window assignment scheduling with resource-dependent processing times to minimise total resource consumption cost. *Int. J. Prod. Res.* **54** (2016) 1–10.
[11] M. Liu, S.J. Wang, F.F. Zheng and C.B. Chu, Algorithms for the joint multitasking scheduling and common due date assignment problem. *Int. J. Prod. Res.* **55** (2017) 6052–6066.
[12] L. Liu, J.J. Wang, F. Liu and M. Liu, Single machine due window assignment and resource allocation scheduling problems with learning and general positional effects. *J. Manuf. Syst.* **43** (2017) 1–14.
[13] G. Mosheiov and D. Oron, Due-window assignment with unit processing-time jobs. *Nav. Res. Logist.* **51** (2004) 1005–1017.
[14] C.H. Papadimitriou and K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity. Englewood Cliffs, NJ, Prentice Hall (1982).
[15] A. Spink, Multitasking information behavior and information task switching: an exploratory study. *J. Doc.* **60** (2004) 336–345.
[16] A. Spink, H.C. Ozmutlu and S. Ozmutlu, Multitasking information seeking and searching processes. *J. Am. Soc. Inform. Sci. Tech.* **53** (2014) 639–652.
[17] D. Wang, Y. Yu, Y. Yin, and T. C. E. Cheng, Multi-agent scheduling problems under multitasking. Int. J. Prod. Res., **59** (2020) 1–31.
[18] Y. Wang, J. Wang, and Y. Yin, Due date assignment and multitasking scheduling with deterioration effect and efficiency promotion. *Comput. Ind. Eng.*, **146** (2020) 106569.
[19] X. Xiong, P. Zhou, Y. Yin, T.C.E. Cheng and D. Li, An exact branch-and-price algorithm for multitasking scheduling on unrelated parallel machines. *Nav. Res. Logist.* **66** (2019) 502–516.
[20] D.L. Yang, C.J. Lai and S.J. Yang, Scheduling problems with multiple due windows assignment and controllable processing times on a single machine. *Int. J. Prod. Econ.*, **150** (2014) 96–103.
[21] Y. Yang, G. Yin, C. Wang, and Y. Yin, Due date assignment and two-agent scheduling under multitasking environment. *J. Comb. Optim.*, **2** (2020).
[22] Y. Yin, T.C.E. Cheng, C.J. Hsu and C.C. Wu, single-machine batch delivery scheduling with an assignable common due window. *Omega* **41** (2013) 216–225.
[23] Y. Yin, T.C.E. Cheng, S.R. Cheng and C.C. Wu, Single-machine batch delivery scheduling with an assignable common due date and controllable processing times. *Comp. Ind. Eng.* **65** (2013) 652–662.
[24] Z. Zhu, F. Zheng, C. Chu, Multitasking scheduling problems with a rate-modifying activity. *Int. J. Prod. Res.* (2016) 1–17.
[25] Z.G. Zhu, M. Liu, C.B. Chu and J.L. Li, Multitasking scheduling with multiple rate-modifying activities. *Int. Trans. Oper. Res.* (2017) 1–21.