



# Multitasking scheduling with batch distribution and due date assignment

Xinrui Xu<sup>1</sup> · Guangqiang Yin<sup>2</sup> · Chunyu Wang<sup>2</sup>

Received: 2 June 2020 / Accepted: 25 July 2020 / Published online: 31 August 2020  
© The Author(s) 2020

## Abstract

This study addresses the multitasking scheduling problems with batch distribution and due date assignment (DDA). Compared with classical scheduling problems with due date-related optimization functions, the job due dates are decision variables rather than given parameters. The jobs completed are distributed in batches, and the sizes of all batches are identical, which may be bounded or unbounded. The jobs in every batch are scheduled one by one. Each batch incurs a fixed cost. Under multitasking environment, it allows the machine to put an uncompleted job on hold and turn to another uncompleted job. The goal is to identify the optimal primary job sequence, the optimal job due dates, and the optimal batch production and distribution strategy so that one of the following two optimization functions is minimised: the total cost composed of the earliness penalty, DDA cost, tardiness penalty and batch distribution cost, and the total cost composed of the earliness penalty, weighted number of late jobs, DDA cost and batch distribution cost. We devise efficient exact algorithms for the problems we consider, and perform numerical experiments to check how multitasking affects the scheduling cost or value, the results of which can assist decision-makers to justify the extent to put to use or refrain from multitasking.

**Keywords** Multitasking · Scheduling · Due date assignment · Batch distribution

## Introduction

The problem we investigate in this study covers three momentous sub-areas of scheduling research, i.e., multitasking scheduling, scheduling with DDA, and batch distribution scheduling. All of these three sub-areas have been widely investigated in the literature. In the remaining part of this section, we briefly review some related research from these sub-areas.

The research about multitasking scheduling is initialized by Hall et al. [12], where the processing of a chosen job can be suspended by other uncompleted jobs. The authors demonstrate that the solution algorithms for some classical scheduling criteria are more complex in terms of time complexity than the corresponding problems without multi-

tasking, and verify the effects of multitasking on the schedule criteria through computational experiments. Subsequently, the research on this line has attracted increasing attention. Hall et al. [13] introduce two different multitasking scheduling problems, in which the first one addresses alternate period processing and the second one investigates shared processing. Ji et al. [15] consider the identical parallel-machine scheduling problem with slack due-window assignment (DWA) under multitasking, in which the slack due windows are machine dependent. Hall et al. [19] consider the multitasking scheduling problem with common DDA such that all the jobs have a common due date. Xiong et al. [24] investigate the unrelated parallel-machine scheduling problem under multitasking, and present an exact mathematical-based programming for the problem with the goal of minimising the total completion time. Li et al. [18] and Wang et al. [23] address multitasking scheduling with two competing agents, and study the computational complexity and present exact algorithms for the problems, in which every agent requires to process its respective jobs and wishes to minimise its respective optimization function that is related to its respective jobs. As for due window assignment scheduling, Zhu et al. [34] address the multitasking scheduling problem with DDA and

✉ Chunyu Wang  
wcy@uestc.edu.cn

<sup>1</sup> School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China

<sup>2</sup> School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China

a rate-modifying activity (RMC), and Zhu et al. [33] further generalize the model to the case with multiple RMCs, in which the RMC may alter the process speed of the machine.

In the aforementioned studies, Ji et al. [15], Liu et al. [19] and Zhu et al. [34] address the due date-related optimization functions, in which the jobs' due dates or time windows require to be decided by the decision-makers in the decision-making process along with the primary job sequence. This kind of scheduling problems is referred to scheduling with DDA or due window assignment in the literature, which has been extensively studied in the area of just-in-time scheduling and has been receiving widespread attention during the past four decades. The literature on this topic is abound, the interested readers are referred to the review papers by Gordon et al. [9,10], and Kaminsky and Hochbaum [11]. Nevertheless, in contrast to our study, all the aforementioned studies as well as the review papers concentrate only on the problem of how to process the jobs while neglecting the problem of scheduling job delivery.

Recently, some researchers pay attention to the study of DDA (due window assignment) and batch distribution scheduling. Chen [4] address the common DDA and batch distribution scheduling, and devise an algorithm with polynomial running time to minimise the weighted sum of tardiness penalty, earliness penalty, batch distribution cost and DDA cost. Shabtay [22] considers the unrestricted DDA and batch distribution scheduling with acceptable lead-times, in which the jobs can be assigned different due dates, and prove that the considered problem is  $NP$ -hard and demonstrate that several special cases may be solved in polynomial time. Yin et al. [29] consider the common DDA and batch distribution scheduling with a RMC to minimise the weighted sum of earliness penalty, holding cost, tardiness penalty, batch distribution cost and DDA cost, and demonstrate that several special cases are polynomial time solvable. Yin et al. [26,28] focus on the common DDA and batch distribution scheduling, where we can reduce the job processing times by assigning a certain amount of resources to process the jobs. Mor and Mosheiov [20] and Yin et al. [30,32] study the DDA and batch distribution scheduling with two competing agents. Yin et al. [27] consider the common DWA and batch distribution scheduling problem such that the due windows of all jobs are identical, which specify the earliest and latest delivery date, and show that the problem with the goal to minimise the weighted sum of earliness cost, window location cost, window size cost, holding cost, tardiness cost, and batch distribution cost is polynomial time solvable. For more results on DDA and batch distribution scheduling, the interested readers are recommended to read the recent papers by Ahmadizar and Farhadi [1], Agnetis et al. [2,3], Gong et al. [8], Kovalyov et al. [16], Li et al. [17], Xu et al. [25] and Yin et al. [31], and the review papers by Chen [5] and Hall and Potts [14].

What is noteworthy is that, as far as our information goes, there is no study addressing the batch delivery schedule in the field of multitasking setting. This study tries to fill up this gap, the contributions of which to the literature can be summarized as follows.

- We introduce a new scheduling model which simultaneously considers batch delivery, multitasking and DDA with the goal to identify the optimal primary job sequence, the optimal job due dates, and the optimal batch production and delivery strategy such that two objective functions considered are minimised.
- We demonstrate that the problem with the first optimization function may be solved in polynomial time and that the problem with the second optimization function is pseudo-polynomial time solvable.
- We perform computational experiments to assess how multitasking affects the scheduling cost or value, the results of which can assist decision-makers to justify the extent to put to use or refrain from multitasking.

We organize the study as follows. The next section formally describes the studied problem, followed by which the preliminary analysis and several properties about the optimal schedule for the studied problems are presented. The subsequent section focuses on developing the solution procedures for solving the problems. The computational experiments to verify the effects of multitasking on the objective function values are shown before the concluding section. The last section provides the conclusions and future studies.

## Problem description

This section formally describes the studied problems. Table 1 summarizes the main notation used throughout this study.

To be precise, a set of  $n$  jobs  $N = \{J_1, \dots, J_n\}$  needs to be scheduled on a machine, which may schedule at most one job at a time. Every job  $J_j$ ,  $j = 1, \dots, n$ , has a non-negative processing time  $p_j$  and a non-negative due date  $d_j$ . Different from the classical assumption that the due date  $d_j$  is pre-defined, it is a decision variable in this study.

As in Hall et al. [12] and Wang et al. [23], we allow uncompleted jobs to suspend the jobs being processing, i.e., it permits the machine to suspend an uncompleted job and turn to another uncompleted job. We regard the job scheduled in anytime as the primary job, and the other uncompleted jobs as the waiting jobs, which may interrupt the processing of the primary job. We assume that each job may be handled as a primary job only once and immediately after another primary job completing its processing.

In the execution of a primary job, two types of times, say interruption time and switching time, are incurred, in which

**Table 1** Notation used in this study

| Notation                       | Meaning  |
|--------------------------------|--|
| $n$                            | The number of jobs   |
| $J = \{J_1, J_2, \dots, J_n\}$ | The set of jobs  |
| $p_j$                          | The processing time of job $J_j$   |
| $d_j$                          | The due date of job $J_j$  |
| $r_k^j$                        | The remaining processing time of job $J_k$ immediately when job $J_j$ becomes a primary job                                |
| $f_k(r)$                       | The remaining processing time of job $J_k$ after interrupting $r, r = 1, \dots, n - 1$ , primary jobs                      |
| $W_j$                          | The set of jobs that are waiting for processing when job $J_j$ becomes a primary job                                       |
| $\varphi(x)$                   | A nondecreasing function on variable $x \in [0, +\infty)$ with $\varphi(0) = 0$  |
| $D_j$                          | The distribution time of job $J_j$ , which equals the completion time of the last job in the batch that contains job $J_j$ |
| $E_j = \max\{d_j - D_j, 0\}$   | The earliness of job $J_j$   |
| $T_j = \max\{D_j - d_j, 0\}$   | The lateness of job $J_j$  |
| $U_j$                          | The lateness indicator, where $U_j = 1$ if $D_j > d_j$ and $U_j = 0$ otherwise   |
| $\alpha \in (0, 1)$            | The interruption rate of all the jobs  |
| $b$                            | The batch size which indicates that any batch can contains at most $b$ jobs  |
| $\vartheta$                    | The distribution cost per batch  |
| $\mu$                          | The unit earliness cost  |
| $\gamma$                       | The unit lateness cost   |
| $\eta$                         | The unit DDA cost  |
| $\omega_j$                     | The cost of job $J_j$ being late   |
| $P = \sum_{j=1}^n p_j$         | The total processing time of all jobs  |

the former represents the time during which the primary job’s waiting jobs interrupt its processing, and the latter measures the time spent in the inspecting its waiting jobs. Specially, given any primary job  $J_j$  and each waiting job  $J_k \in W_j$ , the time that job  $J_k$  suspends the processing of job  $J_j$  is defined as  $\alpha r_k^j$ , which is proportional to the remaining processing

time of job  $J_k$ . The switching time for reviewing the waiting jobs of job  $J_j$  is defined as  $\varphi(|W_j|)$ , which only depends on the number of waiting jobs. As a consequence, it is evident that the remaining processing time of job  $J_k$  after interrupting  $r, r = 1, \dots, n - 1$ , primary jobs equals

$$f_k(r - 1) - \alpha f_k(r - 1) = (1 - \alpha)^r p_k. \tag{1}$$

Thus, the time that job  $J_k$  suspends the  $(r + 1)$ th primary job equals

$$\alpha f_k(r) = \alpha(1 - \alpha)^r p_k. \tag{2}$$

We distribute the completed jobs to the customers in batches, and process the jobs in a batch sequentially. We refer to the sum of the processing times of all the jobs in a batch to the processing time of the batch. The sizes of all batches are identical, which may be bounded or unbounded. By bounded batch size, we mean that there may be at most  $b$  jobs in each batch. In this study, we mainly concentrate on the case with bound batch size, and show how to extend the results to the unbounded case. We distribute each batch to the customers immediately after the last job in it completing its processing, and each distribution incurs a fixed cost  $\vartheta$ . We call the corresponding time as the delivery time of the jobs in the batch.

If we distribute a job to its customer before its due date, an earliness penalty is incurred which depends on how early it is ( $E_j$ ). In addition, if we distribute a job to its customer after its due date, a lateness penalty is incurred, which depends on how late it is ( $T_j$ ) or whether it is late ( $U_j=0$  or  $1$ ). A job is referred to be early if  $U_j = 0$ , and late otherwise.

The goal is to find (1) the optimal primary job sequence, (2) the optimal due dates, (3) the optimal number of batches  $m$ , and (4) the optimal batch distribution partition  $(B_1, B_2, \dots, B_m)$  such that the optimization function

$$F_1 = \sum_{J_j \in N} (\mu E_j + \eta T_j + \gamma d_j) + m\vartheta$$

or

$$F_2 = \sum_{J_j \in N} (\mu E_j + \omega_j U_j + \gamma d_j) + m\vartheta$$

is minimised. In the remaining part of this study, we refer to the problems of minimizing the objective functions  $F_1$  and  $F_2$  as problems  $P1$  and  $P2$ , respectively. In addition, as for problem  $P2$ , we assume that the jobs which would be late are not processed.

The following practical example concerning a two-level supply chain, which involves a steel manufacturer and a set of orders from several customers, motivates the studied problems, where each order consists of producing numbers of

various medium to small steel coils. In this example, the manufacturer is referred to as a single “machine” and the orders are referred to as “jobs”. A sufficient number of transport vehicles are available to distribute the completed orders to the customers, where the transport vehicles have fixed capacity, and the cost per distribution is fixed. The processing requirement dictates that the orders containing in the same batch are processed contiguously and the distribution date of a batch equals the completion time of the last order in the batch. To reduce the setup times that are needed to perform some cleaning operations, or remove a previous container and install a new one when the manufacturer switches processing from one type of steel coil to another type of steel coil, multitasking is permitted, which allows unfinished orders to seize the production resources and suspend the orders under processing. The steel manufacturer will negotiate with customers to set the due dates for completing their orders. To reduce the operating cost and improve the overall satisfaction of the customers, the steel manufacturer requires to determine an effective way to allocate its services over time to perform the orders of the customers in a timely and cost-effective manner. This situation can be modeled as the studied models.

### Structure property analysis

This section provides some structure properties on the optimal solution for problems  $P1$  and  $P2$ .

Given a job sequence, we let  $J_{[j]}$  be the  $j$ th primary job in the sequence, and let  $B_k$  and  $|B_k|$  represent the set of primary jobs scheduled in the  $k$ th batch and the number of primary jobs in  $B_k$  ( $k = 1, 2, \dots$ ), respectively. The following lemma illustrates the property about assignment due dates.

**Proposition 3.1** *There is an optimal solution to the problem so that*

- (i)  $d_j = D_j$  if  $\gamma < \eta$  and  $d_j = 0$  otherwise for all  $j = 1, 2, \dots, n$  when objective function is  $F_1$ ;
- (ii)  $d_j = D_j$  if  $\gamma D_j < \omega_j$  and  $d_j = 0$  otherwise for all  $j = 1, 2, \dots, n$  when the objective function is  $F_2$ .

**Proof** It is similar to the proof of Yin et al. [30]. For completeness, we provide a brief proof for (i). The proof for (ii) is similarly. To be precise, we require to address two cases.

- (1)  $\gamma < \eta$ . Assume there exists a solution  $S$  such that there is a job  $J_j$  with  $d_j = D_j$ , i.e.,  $d_j < D_j$  or  $d_j > D_j$ . In what follows, we prove that a shift of  $d_j$  to the right (resp., left) if  $d_j < D_j$  (resp.,  $d_j > D_j$ ) such that  $d_j = D_j$  can only reduce the optimization function value of  $F_1$ . When  $d_j < D_j$ , the change in the optimization function value of  $F_1$  is  $(\gamma - \eta)(D_j - d_j) < 0$ . On the other hand, when

$d_j > D_j$ , the change in the optimization function value of  $F_1$  is  $-(\gamma + \mu)(d_j - D_j) < 0$ .

- (2)  $\gamma \leq \eta$ . Assume there exists a solution  $S$  such that there is a job  $J_j$  with  $d_j > 0$ . In what follows, we prove that a shift of  $d_j$  to the left such that  $d_j = 0$  can only reduce the optimization function value of  $F_1$ . When  $d_j \geq D_j$ , the change in the optimization function value of  $F_1$  is  $\eta D_j - (\gamma d_j + \mu(d_j - D_j)) \leq 0$ . When  $d_j < D_j$ , the change in the optimization function value of  $F_1$  is  $\eta D_j - (\gamma d_j + \eta(D_j - d_j)) = (\eta - \gamma)d_j \leq 0$ .  $\square$

The proposition above demonstrates that an optimal solution to each of the problems exists where the earliness of each job equals zero. It follows that the optimization functions  $F_1$  and  $F_2$  reduce to

$$\sum_{J_j \in N} \xi D_j + m\vartheta \tag{3}$$

and

$$\sum_{J_j \in N} (\omega_j U_j + \gamma D_j) + m\vartheta, \tag{4}$$

respectively, in which  $\xi = \min\{\gamma, \eta\}$ . As a consequence, it is beneficial to process the primary jobs consecutively from time 0 (exclude the interruption and switching times).

In what follows, we limit our attention to solutions fulfilling the properties above. Now we analyze the formulation of the optimization function  $F_1$  of a given schedule. Given any such primary job sequence  $S$ , which is partitioned into  $m$  batches  $B_1 = \{J_{[1]}, J_{[2]}, \dots, J_{[h_1]}\}, \dots, B_m = \{J_{[h_{m-1}+1]}, J_{[h_{m-1}+2]}, \dots, J_{[n]}\}$ , in which  $h_k$  is the number of primary jobs contained in the first  $k$  batches for  $k = 0, 1, 2, \dots, m$ , with  $h_0 = 0$  and  $h_m = n$ . Thus, for each  $j = 1, \dots, n$ , job  $J_{[j]}$ 's completion time is equal to

$$C_{[j]} = \sum_{l=1}^j p_{[l]} + (1 - (1 - \alpha)^j) \sum_{l=j+1}^n p_{[l]} + \sum_{l=1}^j \varphi(n - l), \tag{5}$$

in which the first term is the sum of the total processing times of the first  $j$  primary jobs, the second term denotes the sum of the interruption times that the last  $n - j$  jobs interrupt the first  $j$  primary jobs, and the third term gives the sum of the switching times during processing the first  $j$  primary jobs.

By Eq. (5), the distribution time  $D_{[j]}$  of job  $J_{[j]}$  such that  $h_{k-1} + 1 \leq j \leq h_k$  for some  $k = 0, 1, \dots, m$  is

$$D_{[j]} = \sum_{l=1}^{h_k} p_{[l]} + (1 - (1 - \alpha)^{h_k}) \sum_{l=h_k+1}^n p_{[l]} + \sum_{l=1}^{h_k} \varphi(n - l), \tag{6}$$

and thus the optimization function  $F_1$  of schedule  $S$  can be formulated as

$$\begin{aligned}
 F_1 &= \sum_{J_j \in N} \xi D_j + m\vartheta \\
 &= \xi \sum_{k=1}^m \sum_{j=h_{k-1}+1}^{h_k} D_{[j]} + m\vartheta \\
 &= \xi \left( \sum_{k=1}^m \sum_{j=h_{k-1}+1}^{h_k} \left( \sum_{l=1}^{h_k} p_{[l]} + (1 - (1 - \alpha)^{h_k}) \right. \right. \\
 &\quad \left. \left. \sum_{l=h_{k+1}}^n p_{[l]} + \sum_{l=1}^{h_k} \varphi(n - l) \right) \right) + m\vartheta \\
 &= \xi \left( \sum_{k=1}^m \left( (n - h_{k-1})P_k + P_k \sum_{l=1}^{k-1} (1 - (1 - \alpha)^{h_l})|B_l| \right. \right. \\
 &\quad \left. \left. + (n - h_{k-1}) \sum_{j=h_{k-1}+1}^{h_k} \varphi(n - j) \right) \right) + m\vartheta, \tag{7}
 \end{aligned}$$

in which  $P_k = \sum_{j=h_{k-1}+1}^{h_k} p_{[j]}$  represents the processing time of batch  $B_k$ .

As for the optimization function  $F_2$ , given any primary job sequence  $S$  with  $n_e$  early jobs, in which the early jobs are partitioned into  $m_e$  batches  $B_1 = \{J_{[1]}, J_{[2]}, \dots, J_{[h_1]}\}, \dots, B_{m_e} = \{J_{[h_{m_e-1}+1]}, J_{[h_{m_e-1}+2]}, \dots, J_{[n_e]}\}$ , then the optimization function  $F_2$  of schedule  $S$  can be formulated as

$$\begin{aligned}
 F_2 &= \sum_{J_j \in N} (\gamma D_j + \omega_j U_j) + m\vartheta \\
 &= \gamma \left( \sum_{k=1}^{m_e} \left( (n - h_{k-1})P_k + P_k \sum_{l=1}^{k-1} (1 - (1 - \alpha)^{h_l})|B_l| \right. \right. \\
 &\quad \left. \left. + (n - h_{k-1}) \sum_{j=h_{k-1}+1}^{h_k} \varphi(n - j) \right) \right) + \sum_{J_j \in L} \omega_j + m\vartheta \tag{8}
 \end{aligned}$$

in which  $L$  stands for the set of late jobs in solution  $S$ .

Based on Eqs. (7) and (8), one can draw the following conclusion.

**Proposition 3.2** *There is an optimal solution to each of the problems  $P1$  and  $P2$  so that the maximum processing time among the primary jobs scheduled in any batch  $B_k$ ,  $k = 1, 2, \dots$ , is equal to or less than the minimum processing time among the primary jobs scheduled in batch  $B_{k+1}$ .*

**Proof** We provide the proof for problem  $P1$ . The proof for problem  $P2$  is analogous. Assume there exists a solution  $S$  with two continuous batches  $B_k$  and  $B_{k+1}$  such that  $p_i > p_j$  for some primary jobs  $J_i \in B_k$  and  $J_j \in B_{k+1}$ . We create a new solution  $S'$  from  $S$  by exchanging primary jobs  $J_i$  and  $J_j$  while keeping the other primary jobs being the same.

According to Eq. (7), the difference between the optimization function values of solutions  $S$  and  $S'$  is

$$\begin{aligned}
 &p_i \left( (n - h_{k-1}) + \sum_{l=1}^{k-1} (1 - (1 - \alpha)^{h_l})|B_l| \right) \\
 &\quad + p_j \left( (n - h_k) + \sum_{l=1}^k (1 - (1 - \alpha)^{h_l})|B_l| \right) \\
 &\quad - \left( p_j \left( (n - h_{k-1}) + \sum_{l=1}^{k-1} (1 - (1 - \alpha)^{h_l})|B_l| \right) \right. \\
 &\quad \left. + p_i \left( (n - h_k) + \sum_{l=1}^k (1 - (1 - \alpha)^{h_l})|B_l| \right) \right) \\
 &= (p_i - p_j) \left( h_k - h_{k-1} - (1 - (1 - \alpha)^{h_k})|B_k| \right) \\
 &= (p_i - p_j)(1 - \alpha)^{h_k}|B_k| > 0.
 \end{aligned}$$

As a consequence, solution  $S'$  is no worse than  $S$ , as required.  $\square$

The proposition above demonstrates that both problems  $P1$  and  $P2$  allow an optimal solution so that the primary jobs are proceeded in terms of the SPT (Shortest Processing Time first) rule. In what follows, we re-arrange the jobs in terms of the SPT rule so that  $d_1 \leq d_2 \leq \dots \leq d_n$ .

### Dynamic programming algorithms

This section devises dynamic programming (DP) algorithms for problems  $P1$  and  $P2$  which relies on the properties given in Lemmas 3.1 and 3.2. Specially, we prove that problem  $P1$  is polynomial time solvable, whereas problem  $P2$  may be solved in pseudo-polynomial time.

Let us begin with devising a polynomial-time forward DP algorithm, denoted as Algorithm  $MDB_T$ , for problem  $P1$ . The procedure of the algorithm is as follows. For each  $j = 0, 1, \dots, n$ , we let  $H_j$  stand for the set of states encoding the partial solutions for the first  $j$  jobs  $J_1, \dots, J_j$ . Each state in  $H_j$  is in the form of  $(a, g)$  such that

- There is exactly  $a$  primary jobs contained in the last batch of the partial solution.
- The optimization function value of the partial solution is  $g$ .

Algorithm  $MDB_T$  begins with an initial tag set  $F_0 = \{(0, 0)\}$ . For each  $j = 1, \dots, n$ , the tag set  $F_j$  is constructed from tag set  $F_{j-1}$ . To be precise, for each tag  $(a, g) \in F_{j-1}$  that corresponds to a partial solution for the jobs  $J_1, \dots, J_{j-1}$ , to create a new solution by appending the next unassigned job  $J_j$  to the end of the above partial

solution, one requires to take the following two cases into account.

*Case T1:* Start a new batch and append job  $J_j$  to it. What is noteworthy is that we take into account the contributions of the jobs to the optimization function value in batches, i.e., the contributions of all jobs contained in a batch are added to the optimization function value of  $F_1$  immediately when we complete the processing of all the jobs contained in the batch. Thus, the contributions of all the jobs contained in the previous batch to the optimization function value of  $F_1$  in this case equals

$$\begin{aligned} & \xi \left( (n - (j - 1 - a)) \sum_{l=j-a}^{j-1} p_l + a \right. \\ & \left. (1 - (1 - \alpha)^{j-1}) \sum_{l=j}^n p_l \right. \\ & \left. + (n - (j - 1 - a)) \sum_{l=0}^{a-1} \varphi(n - j + 1 + l) \right) \end{aligned}$$

according to Eq. (7), in which  $j - 1 - a$  gives the number of primary jobs processed before the previous batch,  $\sum_{l=j-a}^{j-1} p_l$  gives the total processing times of all the jobs contained in the previous batch, and  $\sum_{l=j}^n p_l$  measures the total processing times of all the primary jobs processed after the previous batch. Therefore, the value that increases in the optimization function  $F_1$  due to this operation in this case is

$$\begin{aligned} & \xi \left( (n - (j - 1 - a)) \sum_{l=j-a}^{j-1} p_l + a \right. \\ & \left. (1 - (1 - \alpha)^{j-1}) \sum_{l=j}^n p_l \right. \\ & \left. + (n - (j - 1 - a)) \sum_{l=0}^{a-1} \varphi(n - j + 1 + l) \right) + \vartheta. \end{aligned}$$

As a consequence, one can construct a new tag

$$\begin{aligned} & \left( 1, g + \xi \left( (n - (j - 1 - a)) \sum_{l=j-a}^{j-1} p_l \right. \right. \\ & \left. \left. + a \left( 1 - (1 - \alpha)^{j-1} \right) \sum_{l=j}^n p_l \right. \right. \\ & \left. \left. + (n - (j - 1 - a)) \sum_{l=0}^{a-1} \varphi(n - j + 1 + l) \right) + \vartheta \right) \end{aligned}$$

and add it into  $F_j$ .

*Case T2:* Append job  $J_j$  to the last batch. To ensure the feasibility, there must be  $1 \leq a < b$  by the restriction on the

batch size. By the analysis in Case 1, the value that increases in the optimization function  $F_1$  due to this operation in this case is zero if  $j < n$ , and  $\xi(a+1) \left( \sum_{l=j-a}^j p_l + \sum_{l=0}^a \varphi(l) \right)$  otherwise. Thus, if  $a < b$ , one can construct a new tag  $(a + 1, \tilde{g})$  and add it into  $F_j$ , in which  $\tilde{g} = g$  if  $j < n$  and  $\tilde{g} = g + \xi(a + 1) \left( \sum_{l=j-a}^j p_l + \sum_{l=0}^a \varphi(l) \right)$  otherwise.

What is noteworthy is that not all the generated tags during the procedure above will form a complete better schedule. Thus, it is necessary to find the tags that can be discarded for further consideration by the following dominant rule.

**Lemma 4.1** *Given any tags  $(a, g)$  and  $(a', g')$  in  $F_j$  such that  $a \leq a'$ , and  $g \leq g'$ , one can discard the latter tag.*

**Proof** Given the conditions stated in the lemma, it is evident that every feasible extension of the partial solution corresponding to the latter one is also feasible for the partial solution corresponding to the former one, and leads to a complete solution whose objective function value is equal to or less than that of the solution constructed by using the same extension to the former one.  $\square$

According to the above analyses, we provide the following procedure for Algorithm MDB $_T$ :

**Algorithm MDB $_T$**

Step 1. Re-arrange the jobs in terms of the SPT rule so that  $p_1 \leq \dots \leq p_n$ .

Step 2. Set  $F_0 = \{(0, 0)\}$ .

Step 3. Construct tag set  $F_j$  from  $F_{j-1}$

For  $j = 1$  to  $n$  do

$F_j \leftarrow \emptyset$ ;

For each  $(a, g) \in F_{j-1}$ , do

*/\* Correspond to Case 1 \*/*

$F_j \leftarrow F_j \cup \{(1, g + \xi((n - (j - 1 - a)) \sum_{l=j-a}^{j-1} p_l + a(1 - (1 - \alpha)^{j-1}) \sum_{l=j}^n p_l +$

$(n - (j - 1 - a)) \sum_{l=0}^{a-1} \varphi(n - j + 1 + l)) + \vartheta)\}$ ;

*/\* Correspond to Case 2 \*/*

If  $1 \leq a < b$  then

$F_j \leftarrow F_j \cup \{(a + 1, \tilde{g})\}$ , in which  $\tilde{g} = g$  if  $j < n$  and  $\tilde{g} = g + \xi(a + 1) (\sum_{l=j-a}^j p_l + \sum_{l=0}^a \varphi(l))$  otherwise;

Endif

*/\* Discard nondominated states from  $F_j$  \*/*

Given any tags  $(a, g)$  and  $(a', g)$  in  $F_j$  such that  $a \leq a'$ , discard the latter tag from  $F_j$ ;

Given any tags  $(a, g)$  and  $(a, g')$  in  $F_j$  such that  $g \leq g'$ , discard the latter tag from  $F_j$ ;

Endfor

Step 4. Return the optimal optimization function value  $\min\{g|(a, g) \in F_n\}$ .

**Theorem 4.2** *Problem P1 may be solved by Algorithm  $MDB_T$  with running time  $O(nb)$ .*

**Proof** Due to the fact that all possible scheduling choices for each job  $J_j$  are addressed in Step 3, Algorithm 1 can find an optimal schedule. In Step 1, we execute a sorting procedure requiring  $n \log n$  time. In Step 3, since there are  $b$  different values of  $a$ , there are at most  $b$  different combinations of  $\{a, g\}$  according to the dominant rule. For any state, at most two new states can be constructed. As a consequence, going through  $n$  iterations, Step 3 may be implemented in  $O(nb)$  time, which completes the proof.  $\square$

In what follows, we use the following example to demonstrate Algorithm  $MDB_T$ .

**Example 4.3** Let  $n = 3, \alpha = 0.1, \mu = \eta = 3, \gamma = 1, \vartheta = 10, p_1 = 30, p_2 = 20, p_3 = 15, b = 2$ , and  $\varphi(x) = x$  for all  $x \in [0, +\infty)$ . We implement Algorithm  $MDB_T$  to solve this example in the following way.

Step 1: The jobs are already re-arranged in the LPT order.

Step 2: Set  $F_0 = \{(0, 0)\}$  and  $\xi = \min\{\gamma, \eta\} = 1$ .

Step 3: For  $j = 1$ , we generate a tag  $(1, \vartheta) = (1, 10)$  from the tag  $(0, 0) \in F_0$ . Thus,  $F_1 = \{(1, 10)\}$ .

For  $j = 2$ , we generate two tags  $(1, 10 + 3p_1 + 0.1(p_2 + p_3) + 3\varphi(2) + \vartheta) = (1, 119.50)$  and  $(2, 10)$  from the tag  $(1, 10) \in F_1$ . Thus,  $F_2 = \{(1, 119.50), (2, 10)\}$ .

For  $j = 3$ , we generate two tags  $(1, 119.5 + 2p_2 + (1 - (1 - \alpha)^2)p_3 + 2\varphi(1) + \vartheta) = (1, 174.35)$  and  $(2, 119.50 + 2(p_2 + p_3 + \varphi(1))) = (2, 191.50)$  from the tag  $(1, 119.50) \in F_2$ ; and a tag  $(1, 10 + 3(p_1 + p_2) + 2(1 - (1 - \alpha)^2)p_3 + 3(\varphi(1) + \varphi(2)) + \vartheta) = (1, 184.70)$  from the tag  $(2, 10)$ . During the elimination process,  $(1, 184.70)$  is deleted. Thus,  $F_2 = \{(1, 174.35), (2, 191.50)\}$ , and the optimal solution value is 174.35 and the optimal primary job sequence is  $(J_1, J_2, J_3)$ , where each single job forms a batch.

What is noteworthy is that Algorithm  $MDB_T$  with a slight amendment by deleting the constraint on  $a$  can be applied for solving problem  $P1$  with unbound batch size. In terms of this modification, there are at most  $n$  possible values for  $a$ . Thus, problem  $P1$  with unbounded batch size may be solved in  $O(n^2)$  time.

Now we turn to the solution algorithm for problem  $P2$ . It's worth noting that Algorithm  $MDB_T$  with directly amendments cannot be used to solve problem  $P2$  since we cannot determine which jobs would be scheduled after the current batch. Instead, we devise a backward dynamic programming algorithm, denoted as  $MDB_L$ , that appends a job immediately before the partial schedule. To keep things simple, in the sequel we re-arrange the jobs so that  $p_1 \geq p_2 \geq \dots \geq p_n$ .

Specially, we need to enumerate all possible values of the number of early jobs in the complete optimal solution. Given each such value  $e$  and each  $j, e, j = 0, 1, \dots, n$ , we let  $H_{(e,j)}$  represent the set of tags encoding the partial solutions for jobs  $J_1, \dots, J_j$  so that the number of early jobs in the complete optimal solution is exactly  $e$ . Each tag in  $H_{(e,j)}$  is in the form of  $(x, a, t_1, t_2, g)$  such that

- there is exactly  $x$  primary jobs processed after the first batch of the partial solution,
- there is exactly  $a$  primary jobs in the first batch of the partial solution,
- the total processing time of the primary jobs processed after the first batch of the partial solution is  $t_1$ ,
- the total processing time of the primary jobs processed in the first batch of the partial solution is  $t_2$ ,

in which  $g$  is defined similarly as that in Algorithm  $MDB_T$ .

The algorithm begins with an initial tag set  $H_{(e,0)} = \{(0, 0, 0, 0, 0)\}$ . For each  $j = 1, \dots, n$ , the tag set  $H_{(e,j)}$  is generated from tag set  $H_{(e,j-1)}$ . To be exact, given any tag  $(x, a, t_1, t_2, g) \in F_{(e,j-1)}$  that corresponds to a partial solution for the jobs  $J_1, \dots, J_{j-1}$ , to generate a new schedule by appending the next unassigned job  $J_j$  to the start of the above partial solution, one require to take the following two cases into account.

*Case L1:* assign job  $J_j$  as a late job. The value that increases in the optimization function  $F_2$  due to this operation is  $\omega_j$ . Thus, one can construct a new tag  $(x, a, t_1, t_2, g + w_j)$  and add it into  $H_{(e,j)}$ .

*Case L2:* assign job  $J_j$  as an early job. There must be  $a + x < e$  in this case, and we require to further address two subcases.

*Subcase L21:* Start a new batch and append job  $J_j$  to the new batch. The contributions of all jobs in the previous batch to the objective function value of  $F_2$  in this case is

$$\gamma \left( (x + a)t_2 + a(1 - (1 - \alpha)^{e-x})t_1 + (x + a) \sum_{l=0}^{a-1} \varphi(x + l) \right)$$

by Eq. (8). Thus, the value that increases in the optimization function of  $F_2$  due to this operation in this case equals

$$\gamma \left( (x + a)t_2 + a(1 - (1 - \alpha)^{e-x})t_1 + (x + a) \sum_{l=0}^{a-1} \varphi(x + l) \right) + \vartheta.$$

As a consequence, one can construct a new tag

$$\left( x + a, 1, t_1 + t_2, p_j, g + \gamma((x + a)t_2 + a(1 - (1 - \alpha)^{e-x})t_1 + (x + a) \sum_{l=0}^{a-1} \varphi(x + l)) \right) + \vartheta$$

and add it into  $H_{(e,j)}$ .

*Subcase L22:* Append job  $J_j$  to the first batch. In this case, there must be  $a < b$  by the restriction on the batch size. The value that increases in the optimization function  $F_2$  due to this operation is zero if  $j < n$ , and  $\gamma(x + a) \left( t_2 + \sum_{l=0}^{a-1} \varphi(x + l) \right)$ , otherwise. Thus, if  $a < b$ , one can construct a new tag  $(x, a + 1, t_1, t_2 + p_j, \tilde{g})$  and add it into  $H_{(e,j)}$ , in which  $\tilde{g} = g$  if  $j < n$  and  $\tilde{g} = g + \gamma(x + a) \left( t_2 + \sum_{l=0}^{a-1} \varphi(x + l) \right)$  otherwise.

To cut down the number of states, one can use the following dominant rule.

**Lemma 4.4** *Given any tags  $(x, a, t_1, t_2, g)$  and  $(x', a', t'_1, t'_2, g')$  in  $F_j$  such that  $x \leq x', a \leq a', t_1 \leq t'_1, t_2 \leq t'_2$ , and  $g \leq g'$ , one can discard the latter one.*

**Proof** It is similar to the proof Lemma 3.1. □

According to the analyses above, we provide the following procedure for Algorithm  $MDB_L$ .

**Algorithm  $MDB_L$**

Step 1. Re-arrange the jobs so that  $p_1 \geq \dots \geq p_n$ .  
 Step 2. Let  $H_{(e,0)} = \{(0, 0, 0, 0, 0)\}$  for all  $e = 0, 1, \dots, n$ .  
 Step 3. Construct tag set  $H_{(e,j)}$  from  $H_{(e,j-1)}$   
 For  $e = 0$  to  $n$  do  
   For  $j = 1$  to  $n$  do  
      $H_{(e,j)} \leftarrow \emptyset$ ;  
     For each  $(x, a, t_1, t_2, g) \in H_{(e,j-1)}$ , do  
       /\* Correspond to Case L1 \*/  
        $H_{(e,j)} \leftarrow H_{(e,j)} \cup \{(x, a, t_1, t_2, g + w_j)\}$ ;  
       /\* Correspond to Case L2 \*/  
       If  $x + a < e$  then  
         /\* Correspond to Case L21 \*/  
          $H_{(e,j)} \leftarrow H_{(e,j)} \cup \{(x + a, 1, t_1 + t_2, p_j, g + \gamma((x + a)t_2 + a(1 - (1 - \alpha)^{e-x})t_1 + (x + a) \sum_{l=0}^{a-1} \varphi(x + l)) + \vartheta)\}$ ;

/\* Correspond to Case L22 \*/  
 If  $a < b$  then  
    $H_{(e,j)} \leftarrow H_{(e,j)} \cup \{(x, a + 1, t_1, t_2 + p_j, \tilde{g})\}$ ,  
 in which  $\tilde{g} = g$  if  $j < n$  and  
    $\tilde{g} = g + \gamma(x + a)(t_2 + \sum_{l=0}^{a-1} \varphi(x + l))$  otherwise;  
 Endif  
 Endif  
 Endfor  
 /\* Discard nondominated tags from  $H_{(e,j)}$  \*/  
 Given any tags  $(x, a, t_1, t_2, g)$  and  $(x', a, t_1, t_2, g)$  in  $H_{(e,j)}$  such that  $x \leq x'$ , discard the latter one from  $H_{(e,j)}$ ;  
 Given any tags  $(x, a, t_1, t_2, g)$  and  $(x, a', t_1, t_2, g)$  in  $H_{(e,j)}$  such that  $a \leq a'$ , discard the latter one from  $H_{(e,j)}$ ;  
 Given any tags  $(x, a, t_1, t_2, g)$  and  $(x, a, t'_1, t_2, g)$  in  $H_{(e,j)}$  such that  $t_1 \leq t'_1$ , discard the latter one from  $H_{(e,j)}$ ;  
 Given any tags  $(x, a, t_1, t_2, g)$  and  $(x, a, t_1, t'_2, g)$  in  $H_{(e,j)}$  such that  $t_2 \leq t'_2$ , discard the latter one from  $H_{(e,j)}$ ;  
 Given any tags  $(x, a, t_1, t_2, g)$  and  $(x, a, t_1, t_2, g')$  in  $H_{(e,j)}$  such that  $g \leq g'$ , discard the latter one from  $H_{(e,j)}$ ;  
 Endfor  
 Endfor

Step 4. Return the optimal optimization function value  $\min\{f^A | (x, e - x, t_1, t_2, g) \in H_{(e,n)}, e = 0, 1, \dots, n\}$ .

**Theorem 4.5** *Problem P2 may be solved by Algorithm  $MDB_L$  with running time  $O(n^3bP^2)$ .*

**Proof** It is analogous to the proof of Theorem 3.2, in which the only differences lie in that:  $x, t_1$  and  $t_2$  take at most  $n, P$  and  $P$  different values, respectively. □

Similarly, Algorithm  $MDB_L$  with a slight amendment by deleting the constraint on  $a$  can be applied for solving problem  $P2$  with unbound batch size, which runs in  $O(n^4P^2)$  time.

**Computational experiments**

This section assesses how multitasking affects the scheduling cost or value through computational experiments. In terms of the parameter settings in Hall et al. [12] and Wang et al. [23], we generate different sets of parameter values as follows.

- We selected  $n$  from the set  $\{50, 60, \dots, 120\}$  for problem  $P1$  and from the set  $\{5, 10, 15, 20\}$  for problem  $P2$ .
- We randomly selected  $p_j$  from the uniform distribution  $[10, 50]$ .



- We randomly selected  $\eta$  and  $\gamma$  from the uniform distribution [1, 10].
- We randomly selected  $\vartheta$  from the uniform distribution [20, 100], and  $b$  from the discrete uniform [2,  $n$ ].
- We randomly selected  $\omega_j$  from the uniform distribution [ $bP/n$ ,  $P/2$ ].
- We selected  $\alpha$  from the set {0.01, 0.05, 0.10, 0.15}.

The reason for the setting of  $n$  is as follows: the preliminary computational test indicates that Algorithm  $MDB_T$  takes 2 h on average for solving the instances with  $n = 120$  of problem  $P1$ , and Algorithm  $MDB_I$  takes more 2 h on average for solving the instances with  $n = 20$  of problem  $P2$ . The average running times of the algorithms on testing instances with different numbers of jobs are depicted in Tables 2 and 3, respectively.

Given any combination of the parameters above, we perform two computational experiments, i.e.,  $\varphi(x) = 0.05x$  and  $\varphi(x) = -0.05x$ . And for any combination of  $n$  and  $\alpha$ , 30 instances were randomly chosen and the average results were reported.

We apply the algorithm for each problem by setting  $\alpha = 0$  for solving the corresponding problem without multitasking. The developed DP algorithms are coded in MATLAB and the numerical experiments are performed on a notebook computer with a 3.80-GHz CPU and 16 GB memory.

Let  $F_1$  and  $F'_1$  be the optimal optimization function values to the multitasking problem  $P1$  and the corresponding problem without multitasking, respectively, and  $LT_{avg}$  be the mean value of  $\frac{F_1 - F'_1}{F'_1}$  over the 50 chosen instances for any combination of  $n$  and  $\alpha$ , which demonstrates the average multitasking cost or value. In a similar way, we let  $F_2$  and  $F'_2$  be the optimal optimization function values to the multitasking problem  $P2$  and the corresponding problem without multitasking, respectively, and  $LL_{avg}$  be the mean value of  $\frac{F_2 - F'_2}{F'_2}$  over the 50 chosen instances.

The results for problems  $P1$  and  $P2$  are summarized in Tables 2 and 3, from which we have the following observations.

- As expected, the affect of multitasking on the scheduling cost or value increases as the  $\alpha$  value increases, especially for problem  $P1$ . For instances, when  $\varphi(x) = 0.05x$ , the  $LT_{avg}$  value for  $\alpha = 0.15, 0.10$  and  $0.05$  respectively increases 54.02%, 39.12% and 28.17% on average compared to that for  $\alpha = 0.01$  for problem  $P1$ , and the  $LL_{avg}$  value for  $\alpha = 0.15, 0.10$  and  $0.05$  respectively increases 512.17%, 158.37% and 158.71% on average compared to that for  $\alpha = 0.01$  for problem  $P2$ . In addition, when  $\varphi(x) = -0.05x$ , the  $LT_{avg}$  value for  $\alpha = 0.15, 0.10$  and  $0.05$ , respectively increases 96.81%, 84.05% and 43.47% on average compared to that for  $\alpha = 0.01$  for problem

$P1$ , and the  $LL_{avg}$  value for  $\alpha = 0.15, 0.10$  and  $0.05$ , respectively, increases 301.63%, 196.56% and 112.32% on average compared to that for  $\alpha = 0.01$  for problem  $P2$ . The reason behind this is that the interruption time that the waiting jobs suspend the primary jobs becomes large when  $\alpha$  increases.

- The affect of the number of jobs on the  $LT_{avg}$  and  $LL_{avg}$  values is obviously. From Tables 1 and 2, we can see that the instances with more larger  $n$  induce more larger cost of multitasking for most problem instances. For instance, the average cost of multitasking ranges from 98.15 to 143.61% when  $(\alpha, \varphi(x)) = (0.10, -0.05x)$  and ranges from 98.50 to 116.67% when  $(\alpha, \varphi(x)) = (0.10, -0.05x)$ , which increases approximately in proportion to the number of jobs.
- Negative  $LT_{avg}$  or  $LL_{avg}$  value is possible, implying that we can get some revenue from multitasking. For instance, when  $n = 5$ , we can get 0.01% revenue from multitasking when  $(\alpha, \varphi(x)) = (0.05, -0.05x)$ , and get 0.05% revenue from multitasking when  $(\alpha, \varphi(x)) = (0.01, -0.05x)$  for problem  $P2$ . However, this result does not continue as the number of jobs increases.
- The affect of multitasking for problem  $P1$  is more larger than that for problem  $P2$ . This is due to the fact that multitasking will affect the completion times of jobs, and the optimization function of problem  $P1$  is more related to the completion times of jobs compared to that of problem  $P2$ .

## Conclusions

In this study, we investigate a novel scheduling model that coinstantaneously involves batch distribution, multitasking and DDA. In the developed model, the job due dates are to be decided by the decision-makers in the decision-making process. The completed jobs are distributed to their customers in batches, and the sizes of all batches are identical, which may be bounded or unbounded. The jobs contained in each batch are scheduled sequentially. In addition, it permits the machine to suspend an uncompleted job and turn to another uncompleted job. The goal is to identify the optimal primary job sequence, the optimal job due dates, and the optimal batch production and delivery strategy so that one of the following optimization functions are minimised: the total cost composed of the earliness penalty, DDA cost, tardiness penalty and batch distribution cost, and the total cost composed of the earliness penalty, weighted number of late jobs, DDA cost and batch distribution cost. We prove that the problem with the first optimization function may be solved in polynomial time and that the problem with the second optimization function is pseudo-polynomial time solvable. We also con-

**Table 2** The results for problem P1

| <i>n</i> | Avg. time (s) | Cost of multitasking under $(\alpha, \varphi(x))$ |                   |                   |                   |                    |                    |                    |                    |  |  |  |
|----------|---------------|---|-------------------|-------------------|-------------------|--------------------|--------------------|--------------------|--------------------|--|--|--|
|          |               | (0.15, 0.05x) (%)                                 | (0.10, 0.05x) (%) | (0.05, 0.05x) (%) | (0.01, 0.05x) (%) | (0.15, -0.05x) (%) | (0.10, -0.05x) (%) | (0.05, -0.05x) (%) | (0.01, -0.05x) (%) |  |  |  |
| 50       | 28.01         | 112.61  | 98.15             | 88.50             | 64.31             | 107.23             | 98.50              | 70.46              | 52.15              |  |  |  |
| 60       | 34.36         | 116.10  | 103.05            | 95.35             | 69.94             | 109.51             | 105.14             | 76.13              | 54.21              |  |  |  |
| 70       | 491.22        | 130.22  | 112.96            | 104.58            | 83.12             | 114.18             | 108.65             | 85.41              | 55.26              |  |  |  |
| 80       | 975.08        | 131.51  | 123.64            | 109.61            | 87.86             | 116.93             | 107.12             | 87.24              | 60.32              |  |  |  |
| 90       | 1059.74       | 139.92  | 126.78            | 121.74            | 94.26             | 119.16             | 113.51             | 91.02              | 62.15              |  |  |  |
| 100      | 1372.05       | 144.21  | 131.54            | 125.16            | 101.34            | 123.38             | 115.26             | 90.35              | 63.98              |  |  |  |
| 110      | 1555.82       | 147.29  | 139.92            | 126.30            | 105.21            | 126.19             | 114.98             | 96.81              | 68.22              |  |  |  |
| 120      | 1851.00       | 151.34  | 143.61            | 132.71            | 116.34            | 130.54             | 116.67             | 97.42              | 70.13              |  |  |  |

**Table 3** The results for problem P2

| <i>n</i> | Avg. time (s) | Cost of multitasking under $(\alpha, \varphi(x))$ |                   |                   |                   |                    |                    |                    |                    |  |  |  |
|----------|---------------|---|-------------------|-------------------|-------------------|--------------------|--------------------|--------------------|--------------------|--|--|--|
|          |               | (0.15, 0.05x) (%)                                 | (0.10, 0.05x) (%) | (0.05, 0.05x) (%) | (0.01, 0.05x) (%) | (0.15, -0.05x) (%) | (0.10, -0.05x) (%) | (0.05, -0.05x) (%) | (0.01, -0.05x) (%) |  |  |  |
| 5        | 10.02         | 0.16  | 0.04              | 0.05              | 0.01              | 0.04               | 0.00               | -0.01              | -0.05              |  |  |  |
| 10       | 130.66        | 2.98  | 2.05              | 1.49              | 0.83              | 1.08               | 0.77               | 0.44               | 0.12               |  |  |  |
| 15       | 1175.31       | 7.32  | 5.94              | 5.68              | 3.21              | 6.13               | 4.01               | 3.38               | 1.23               |  |  |  |
| 20       | 6870.21       | 14.18   | 10.92             | 9.67              | 5.42              | 9.48               | 7.19               | 6.18               | 3.29               |  |  |  |

duct numerical experiments on randomly chosen instances to assess how multitasking affects the scheduling cost or value, the results of which can assist decision-makers to justify the extent to put to use or refrain multitasking.

As for future research, the following topics are interesting and necessary.

- Providing the computational complexity status of the problem  $P2$ , i.e. whether it is  $NP$ -hard.
- Investigating the model with other due date assignment methods.
- Extending the model to other machine setting, for example, flowshop or parallel-machine setting.
- Investigating the model in a uncertain or dynamic environment, for example in the present of random processing times, unexpected machine breakdown, etc.
- Extending the model to the case with energy efficiency, consumption, or cost as constraints or objectives (Gao et al. [7]).
- Developing effective intelligent optimization algorithms for solving large-scale problem, such as Tabu search, genetic algorithm, or genetic programming (Nguyen et al. [21]).

**Acknowledgements** We thank the editor, associate editor, and two anonymous referees for their helpful comments on earlier versions of our paper. This paper was supported in part by the National Natural Science Foundation of China under grant number 71971041, by the Outstanding Young Scientific and Technological Talents Foundation of Sichuan Province under grant number 2020JDJQ0035, and in part by the Science and Technology Project of Education Department of Jiangxi Province under grant number 180375.

## Compliance with ethical standards

**Conflict of interest** On behalf of all authors, the corresponding author states that there is no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Ahmadizar F, Farhadi S (2015) Single-machine batch delivery scheduling with job release dates, due windows and earliness, tardiness, holding and delivery costs. *Comput Oper Res* 53:194–205
2. Agnetis A, Aloulou MA, Kovalyov MY (2017) Integrated production scheduling and batch delivery with fixed departure times and inventory holding costs. *Int J Prod Res* 55(20):6193–6206
3. Agnetis A, Aloulou MA, Fu LL (2016) Production and interplant batch delivery scheduling: dominance and cooperation. *Int J Prod Econ* 182:38–49
4. Chen ZL (1996) Scheduling and common due date assignment with earliness-tardiness penalties and batch delivery costs. *Eur J Oper Res* 93:49–60
5. Chen ZL (2010) Integrated production and outbound distribution scheduling: review and extensions. *Oper Res* 58(1):130–148
6. Cheng BY, Leung JYT, Li K, Yang SL (2015) Single batch machine scheduling with deliveries. *Naval Res Logist* 62:470–482
7. Gao K, Huang Y, Sadollah A, Wang L (2020) A review of energy-efficient scheduling in intelligent production systems. *Complex Intell Syst* 6:237–249
8. Gong H, Tang L, Leung JYT (2016) Parallel machine scheduling with batch deliveries to minimize total flow time and delivery cost. *Naval Res Logist* 63:492–502
9. Gordon V, Proth JM, Chu C (2002) A Survey of the state-of-the-art of common due date assignment and scheduling research. *Eur J Oper Res* 139:1–25
10. Gordon V, Strusevich V, Dolgui A (2012) Scheduling with due date assignment under special conditions on job processing. *J Sched* 15:447–456
11. Kaminsky P, Hochbaum D (2014) Due-date quotation models and algorithms. In: Leung JY-T (ed) *Handbook of scheduling: algorithms, models and performance analysis*. CRC Press, Boca Raton, pp 20:1–20:22
12. Hall NG, Leung JYT, Li CL (2015) The effects of multitasking on operations scheduling. *Prod Oper Manag* 24:1248–1265
13. Hall NG, Leung JYT, Li CL (2016) Multitasking via alternate and shared processing: algorithms and complexity. *Discrete Appl Math* 208:41–58
14. Hall NG, Potts CN (2003) Supply chain scheduling: batching and delivery. *Oper Res* 51:566–584
15. Ji M, Zhang W, Liao L, Cheng TCE, Tan Y (2019) Multitasking parallel-machine scheduling with machine-dependent slack due-window assignment. *Int J Prod Res* 57:1667–1684
16. Kovalyov MY, Oulamara A, Soukhal A (2015) Two-agent scheduling with agent specific batches on an unbounded serial batching machine. *J Sched* 18:423–434
17. Li F, Chen ZL, Tang L (2017) Integrated production, inventory and delivery problems: complexity and algorithms. *INFORMS J Comput* 29(2):232–250
18. Li S, Chen R, Tian J (2019) Multitasking scheduling problems with two competitive agents. *Eng Optim*. <https://doi.org/10.1080/0305215X.2019.1678609>
19. Liu M, Wang SJ, Zheng FF, Chu CB (2017) Algorithms for the joint multitasking scheduling and common due date assignment problem. *Int J Prod Res* 55:6052–6066
20. Mor B, Mosheiov G (2011) Single machine batch scheduling with two competing agents to minimize total flowtime. *Eur J Oper Res* 215:524–531
21. Nguyen S, Mei Y, Zhang M (2017) Genetic programming for production scheduling: a survey with a unified framework. *Complex Intell Syst* 3:41–66
22. Shabtay D (2010) Scheduling and due date assignment to minimize earliness, tardiness, holding, due date assignment and batch delivery costs. *Int J Prod Econ* 123:235–242
23. Wang D, Yu Y, Yin Y, Cheng TCE (2020) Multi-agent scheduling problems under multitasking. *Int J Prod Res*. <https://doi.org/10.1080/00207543.2020.1748908>
24. Xiong X, Zhou P, Yin Y, Cheng TCE, Li D (2019) An exact branch-and-price algorithm for multitasking scheduling on unrelated parallel machines. *Naval Res Logist* 66:502–516

25. Xu S, Bean JC (2016) Scheduling parallel-machine batch operations to maximize on-time delivery performance. *J Sched* 19:583–600
26. Yin Y, Cheng TCE, Cheng SR, Wu CC (2013) Single-machine batch delivery scheduling with an assignable common due date and controllable processing times. *Comput Ind Eng* 65:652–662
27. Yin Y, Cheng TCE, Hsu CJ, Wu CC (2013) Single-machine batch delivery scheduling with an assignable common due window. *Omega* 41:216–225
28. Yin Y, Cheng TCE, Wu CC, Cheng SR (2013) Single-machine common due-date scheduling with batch delivery costs and resource-dependent processing times. *Int J Prod Res* 51(17):5083–5099
29. Yin Y, Cheng TCE, Xu D, Wu CC (2012) Common due date assignment and scheduling with a rate-modifying activity to minimize the due date, earliness, tardiness, holding, and batch delivery cost. *Comput Ind Eng* 63:223–234
30. Yin Y, Li D, Wang D, Cheng TCE (2018) Single-machine serial-batch delivery scheduling with two competing agents and due date assignment. *Ann Oper Res*. <https://doi.org/10.1007/s10479-018-2839-6>
31. Yin Y, Wang Y, Cheng TCE, Wang D, Wu CC (2016) Two-agent single-machine scheduling to minimize the batch delivery cost. *Comput Ind Eng* 92:16–30
32. Yin Y, Yang Y, Wang D, Cheng TCE, Wu CC (2018) Integrated production, inventory, and batch delivery scheduling with due date assignment and two competing agents. *Naval Res Logist* 65:393–409
33. Zhu ZG, Liu M, Chu CB, Li JL (2019) Multitasking scheduling with multiple rate-modifying activities. *Int Trans Oper Res* 26:1956–1976
34. Zhu ZG, Zheng FF, Chu CB (2017) Multitasking scheduling problems with a rate-modifying activity. *Int J Prod Res* 55:296–312

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.