Journal of Scheduling (2021) 24:209–221
https://doi.org/10.1007/s10951-020-00668-1

Scheduling coupled tasks with exact delays for minimum total job completion time

Bo Chen[1] · Xiandong Zhang[2]

Accepted: 12 September 2020 / Published online: 3 November 2020

Abstract
In this paper, we fill in a conspicuous gap in research on scheduling coupled tasks. We draw a full complexity picture for single-machine scheduling of coupled tasks with exact time delays in between with the objective of minimizing the total of job completion times.

Keywords Scheduling · Coupled task · Exact delay

1 Introduction

The problem of scheduling coupled tasks with exact delays was first introduced by Shapiro (1980), who modeled the scheduling of a radar tracking system in which the system transmits pulses and receives their reflections once every specified update period. The system cannot transmit a pulse at the same time that a reflected pulse is arriving. The transmission and reception of a radar pulse are modeled as a pair of coupled tasks with a fixed length of time lag in between. This basic model of scheduling coupled tasks has been studied for various radar tracking systems of multiple targets and/or multiple functionalities with various objective functions (Elshafei et al. 2004; Farina and Neri 1980; Izquierdo-Fuente and Casar-Corredera 1994; Orman et al. 1996, 1998). This scheduling model has also found many other applications, ranging from improving performance of submarine torpedoes to chemistry manufacturing processes, to scheduling systems of robotic cells and to patient appointments in a chemotherapy outpatient clinic, as documented by Khatami et al. (2020). Various solution approaches have been proposed. Elshafei et al. (2004) presented integer programming models for the multi-target tracking system. More recently, Zhang et al. (2019a) developed a hybrid genetic algorithm for scheduling a phased array radar, and Zhang et al. (2019b) provided an online algorithm for a radar system having multiple inputs and multiple outputs.

It has been about 40 years since the problem of scheduling coupled tasks was introduced to the scheduling community. As pointed out recently by Khatami et al. (2020) in their survey, the early years witnessed only a few studies on this topic, and the majority of works have been developed in the last few years. In concluding their survey, Khatami et al. (2020) state that “there has been no published research investigating the single-machine setting with an objective function other than the makespan, except for those in the cyclic setting.” They place “computational complexity under different objective functions” first in their list of suggested future directions. In this paper, we address this very issue and draw a full complexity picture for the problems of single-machine scheduling of coupled tasks, with the objective of minimizing the total of job completion times.

In what follows, we start with a brief literature review in Sect. 2 on the studies related to complexity status of coupled task scheduling problems, followed in Sect. 3 by a detailed description of the scheduling problem under investigation in this paper, together with a full complexity picture we complete in this paper. In Sects. 4 and 5 we present technical details of our complexity picture to identify NP-hard problems and polynomially solvable problems, respectively. Finally, we provide some concluding remarks in Sect. 6.

✉ Xiandong Zhang
xiandongzhang@fudan.edu.cn

Bo Chen
b.chen@warwick.ac.uk

1 Warwick Business School, University of Warwick, Coventry CV4 7AL, UK

2 School of Management, Fudan University, Shanghai 200433, China

2 Literature review

In this section, we briefly review the literature of studies related to complexity status of coupled task scheduling problems. We adopt the standard three-field representation $\alpha|\beta|\gamma$ introduced by Graham et al. (1979) for a machine scheduling problem, where α, β and γ represent, respectively, the machine environment, job characteristics and objective function. For example, we denote by $1|(a_j, L_j, b_j)|C_{\max}$ the single-machine scheduling problem of coupled tasks with the objective of minimizing the makespan, where each job j consists of a pair of tasks of processing times a_j and b_j, respectively, and with an exact time delay L_j between the completion time of its first task and the start time of its second task.

2.1 Single-machine problems

Shapiro (1980) pointed out that the basic scheduling problem $1|(a_j, L_j, b_j)|C_{\max}$ is NP-hard, a view strengthened by Orman and Potts (1997), who established that even the restricted problems $1|(p_j, p_j, p_j)|C_{\max}$, $1|(a_j, L, b)|C_{\max}$, and $1|(p, L_j, p)|C_{\max}$ are all strongly NP-hard, and by Yu et al. (2004), who proved that problem $1|(1, L_j, 1)|C_{\max}$ is strongly NP-hard. The view was further strengthened by Condotta and Shakhlevich (2012), who demonstrated that the more restricted problem $1|(1, L_j, 1, \pi_a)|C_{\max}$ is already strongly NP-hard, where π_a means that the processing sequence of the first tasks of all jobs is fixed. On the other hand, Orman and Potts (1997) and Hwang and Lin (2011) provided polynomial-time algorithms for various other special cases of problem $1|(a_j, L_j, b_j)|C_{\max}$. Consequently, the complexity picture for single-machine scheduling problems with the objective of minimizing the makespan has been completed. In this paper, we draw a *new* and *complete* picture for the complexity of the single-machine scheduling problems with a different objective, which is to minimize the total of job completion times.

2.2 Shop problems

Let us review the literature on scheduling coupled tasks with more than one machine. Studies have been focused on the environment of two-machine *flow shops*, denoted by $F2$, in which the two tasks of each job need to be processed on two dedicated machines in the same machine sequence. As pointed out by Khatami et al. (2020), problem $F2|(1, L_j, 1)|C_{\max}$ was shown by Yu et al. (2004) to be strongly NP-hard, even if only two distinct values are used for the exact delays of the coupled tasks. Problem $F2|(a_j, L, b_j)|\sum C_j$ was proved by Leung et al. (2007) to be strongly NP-hard. Some polynomially solvable cases were also provided by Leung et al. (2007) for problems $F2|(a, L, b, a \geq b)|\sum C_j$ and $F2|(a, L, b, a < b)|\sum C_j$. Huo et al. (2009) provided several polynomial-time algorithms for multiple variants of the two-machine flow-shop problem of minimizing the total of job completion times.

2.3 Additional constraints and more

Simonin et al. (2011) generalized the standard scheduling problems of coupled tasks by introducing the so-called compatibility constraints when they studied the problem of scheduling submarine torpedoes. In this scenario, two jobs are *compatible* if and only if a task of one job can be processed during the time lag of the other job. They established some complexity results with approximation algorithms. Bessy and Giroudeau (2019) considered the scheduling problem with compatibility constraints with the objective of maximizing the number of jobs that can be completed by a given due date.

Recently, Khatami et al. (2020) provided an updated survey of research on scheduling coupled tasks with exact time delays. An earlier survey was given by Blazewicz et al. (2012). We refer the reader to these two reviews for more details.

3 Problem description and preliminaries

Our generic problem can be stated as follows in terms of machine scheduling terminology. There is a set $\mathcal{J} = \{1, \ldots, n\}$ of n independent jobs and they need to be scheduled on a single machine that is continuously available from the start. Each job $j \in \mathcal{J}$ consists of two operations $\{O_{j1}, O_{j2}\}$, which require processing on the single machine for an uninterrupted a_j and b_j time units, respectively. There should be an exact time delay of L_j between the completing of operation O_{j1} and the starting of operation O_{j2}. The single machine can process no more than one operation at any time. Our objective is to find a feasible schedule that minimizes the total of job completion times, where the completion time of a job $j \in \mathcal{J}$ is when its second operation O_{j2} is completed.

When we say a job is scheduled *before* another job, we mean that both operations of the former job are scheduled before the latter job starts. We say job j is *interleaved with* job k if the first operation of job k is processed in the time lag of job j (regardless of when the second operation of job k is processed), or more generally we simply say these two jobs are interleaved with each other. Given a schedule σ, a *block* of σ is a partial schedule σ' of a subset $\mathcal{J}' \subseteq \mathcal{J}$ of jobs such that (a) it includes both operations of each job $j \in \mathcal{J}'$ and (b) both operations of any job $k \in \mathcal{J} \backslash \mathcal{J}'$ are completed before the starting time s of the first operation in σ', or are started after the completion time t of the last operation in σ'. The *length* of the block σ' is defined as $t - s$.

Table 1 Summary of new results

Problems $1\|(\cdots)\|\sum C_j$	Complexity	Location in the paper
(p, L_j, p)	Strongly NP-hard	Theorem 1
(a, L_j, b)	Strongly NP-hard	✓
(a, L_j, b_j)	Strongly NP-hard	✓
(a_j, L_j, b)	Strongly NP-hard	✓
(a_j, L_j, b_j)	Strongly NP-hard	✓
(a_j, L, b)	Strongly NP-hard	Theorem 2
(a, L, b_j)	Strongly NP-hard	Theorem 3
(a_j, L, b_j)	Strongly NP-hard	✓
(p_j, p_j, p_j)	Strongly NP-hard	Theorem 4
(p_j, p_j, b_j)	Strongly NP-hard	✓
(a_j, p_j, p_j)	Strongly NP-hard	✓
(p_j, L_j, p_j)	Strongly NP-hard	✓
(p, p, b_j)	Polynomially solvable	Theorem 5
(p, p, b)	Polynomially solvable	✓
(p, p, p)	Polynomially solvable	✓
(a_j, p, p)	Polynomially solvable	Theorem 6
(a, p, p)	Polynomially solvable	✓
(p, L, p)	Polynomially solvable	Theorem 7
(a, L, b)	Polynomially solvable[a]	Theorem 8

[a] where a, L and b are fixed values

Hence, the *length* of job $j \in \mathcal{J}$ viewed as a block is equal to $a_j + L_j + b_j$. Since we wish to minimize the total job completion time, we assume without loss of generality that all blocks are scheduled contiguously without machine idle time in between. If block σ' viewed as a (full) schedule of jobs $\mathcal{J}'$ contains no block other than itself, then we say it is a *minimal* block. In this paper, a block is assumed to be minimal.

Given any schedule, let us use n_k to denote the number of jobs processed in block k and ℓ_k the length of the block. Define the *density* of block k as $\Delta_k = n_k/\ell_k$. It is easy to verify the following lemma, which we call the density property of optimal schedules.

Lemma 1 (Density Property) *Given any optimal schedule, if blocks k_1 and k_2 are adjacent, with the former immediately before the latter in any optimal schedule, then we must have $\Delta_{k_1} \geq \Delta_{k_2}$.*

In the next two sections, we determine the computational complexity of all special cases of our general problem $1|(a_j, L_j, b_j)|\sum C_j$, in terms of imposing restrictions on the three parameters a_j, b_j and L_j if they are in the form of constants. Our findings can be summarized in Table 1, where (and thereafter) if more than one parameter of $\{a_j, L_j, b_j\}$ share a common value, we use p or p_j to denote the value, and where a symbol "✓" in the last column indicates that the corresponding result is implied by the one immediately above with an explicit location in the paper. The relationships between different problems can be illustrated in Fig. 1, where $A \to \bar{A}$ indicates that problem A is a special case of problem $\bar{A}$, and the dotted circle and two dotted arrows correspond to the footnote in Table 1.

In establishing strong NP-hardness, we use a reduction from a strong NP-complete problem, known as 3-PARTITION, which is stated as follows: given a non-empty finite set $Q = \{e_i : i \in I = \{1, \ldots, 3q\}\}$ of $3q$ positive integers and another positive integer E, such that $E/4 < e_i < E/2$ for all $i \in I$ and $\sum_{i \in I} e_i = qE$, can I be partitioned into q disjoint subsets $I_1, \ldots, I_q$ such that $\sum_{i \in I_k} e_i = E$ for all $k = 1, \ldots, q$?

4 Proofs of NP-hardness

In this section, we identify all NP-hard problems. It turns out that all these problems are actually strongly NP-hard. We establish strong NP-hardness in each subsection for a problem based on a reduction from 3- PARTITION to the corresponding decision version of the problem.

4.1 Problem $1|(p, L_j, p)|\sum C_j$

This subsection establishes the following result.

Theorem 1 *Problem $1|(p, L_j, p)|\sum C_j$ is strongly NP-hard.*

Given an instance of 3- PARTITION as specified in Sect. 3, we construct an instance $\mathcal{I}_1$ of the decision version of prob-

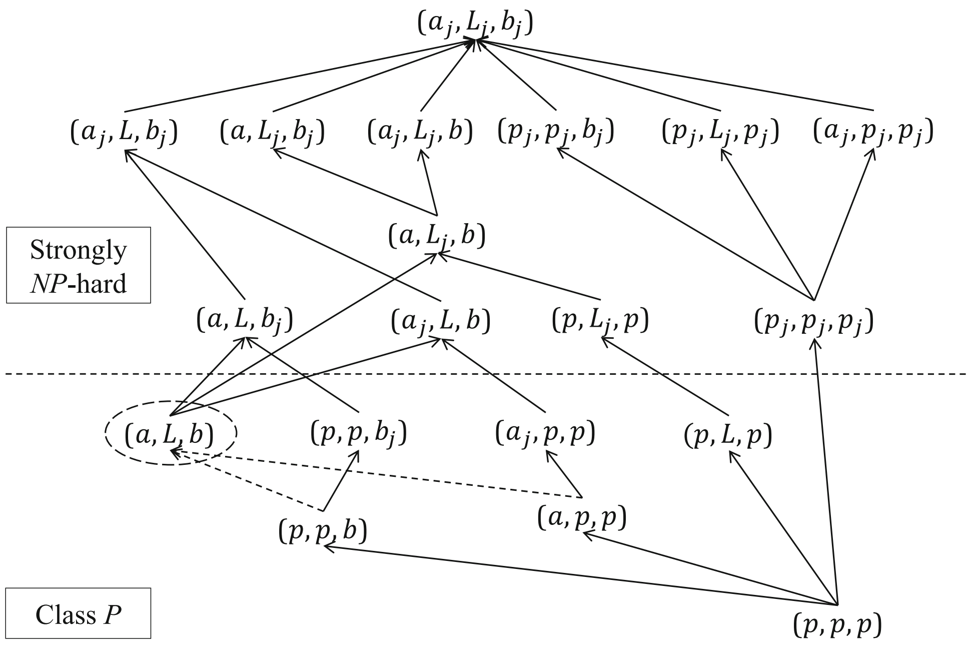

Fig. 1 Relationships between problems

lem $1|(p, L_j, p)|\sum C_j$. Denote $\omega = 40q(2q+1)E$, let the common operation size $p = 2E$ and define $\omega + 6q$ jobs in three sets as follows: $3q$ small jobs in $\mathcal{J}_1$ as *partition* jobs, q large jobs in $\mathcal{J}_2$ as *division* jobs, and $\omega + 2q$ medium jobs in $\mathcal{J}_3$ as *accessory* jobs. More specifically,

$$
\begin{array}{ll}
\mathcal{J}_1 = \{J_j : L_j = e_j,\ j = 1, \ldots, 3q\}, & \text{(partition jobs)} \\
\mathcal{J}_2 = \{J_j : L_j = 13E,\ j = 3q+1, \ldots, 4q\}, & \text{(division jobs)} \\
\mathcal{J}_3 = \{J_j : L_j = 2E-1,\ j = 4q+1, \ldots, \omega + 6q\}. & \text{(accessory jobs)}
\end{array}
$$

We show that there is a 3- Partition solution if and only if there exists a feasible schedule with total job completion time $\sum C_j < y$, where $y = \omega + \alpha$, $\alpha = \beta\omega + (6E-1)\omega(\omega+1)/2$ with $\beta = q(29E-2)$.

Suppose 3- Partition has a solution such that disjoint index sets $\{I_k : k = 1, \ldots, q\}$ satisfy $\sum_{j \in I_k} e_j = E$ $(k = 1, \ldots, q)$. Let us construct the following partial schedule of jobs in q blocks. Each block k $(1 \leq k \leq q)$ consists of one large job (of $\mathcal{J}_2$) with three small jobs of indices in I_k scheduled contiguously between the two operations of the large job, leaving the machine idle for three time lags $\{e_j : j \in I_k\}$. Therefore, each block has a fixed length of $17E$. Following these q blocks without machine idle in between, we schedule all medium jobs (of $\mathcal{J}_3$) contiguously with $2E-1$ machine idle time between the two operations of each job. We illustrate the whole schedule in Fig. 2.

The total completion time of the q large jobs is $17\frac{q(q+1)}{2}E$, that of the $3q$ small jobs in the q blocks is less than $3 \times 17\frac{q(q+1)}{2}E$, that of the first $2q$ medium jobs is equal to $34q^2E + \frac{2q(2q+1)}{2}(6E-1)$ with the last one finishing at time β, and that of the last ω medium jobs is $\beta\omega + \frac{\omega(\omega+1)}{2}(6E-1) = \alpha$. Since $17\frac{q(q+1)}{2}E + 3 \times 17\frac{q(q+1)}{2}E + 34q^2E + \frac{2q(2q+1)}{2}(6E-1) = \omega - q(2q+1)$, the total completion time of all jobs is less than $\omega + \alpha$.

Now let us assume that 3- Partition has no solution and show that any optimal schedule σ of the $6q + \omega$ jobs will have a total job completion time $\sum C_j(\sigma)$ at least y.

Given an optimal schedule σ. Because in any feasible schedule, medium and small jobs cannot be interleaved and a large job can have at most two medium jobs between its two operations, there are at least ω medium jobs that are scheduled as single-job blocks in optimal schedule σ, each of which has a density of $1/(6E-1)$. Hence, we can assume that all these ω single-job blocks are scheduled consecutively to form a composite block B. Let us consider two cases.

Case 1: the densities of all other blocks in σ are at least $1/(6E-1)$. Then all these blocks can be scheduled before the composite block B in an optimal schedule. Since these blocks contain $3q$ small jobs, q large jobs and $2q$ medium jobs, the total machine time these $6q$ jobs require is strictly larger than β since 3- Partition has no solution due to the following facts: in any feasible schedule, (a) all $3q$ small jobs have a total length of $13qE$ and all the $2q$ medium jobs have a total length of $2q(6E-1)$; (b) the q large jobs have a total processing time of $4qE$. Therefore, these $6q$ jobs will occupy at least $\beta = 13qE + 2q(6E-1) + 4qE$ machine time. However, when 3- Partition has no solution, the time lags of the large jobs cannot be fully filled, i.e., there is at least one large job whose time lag of $13E$ cannot be fully occupied by small or medium jobs inclusive of their own

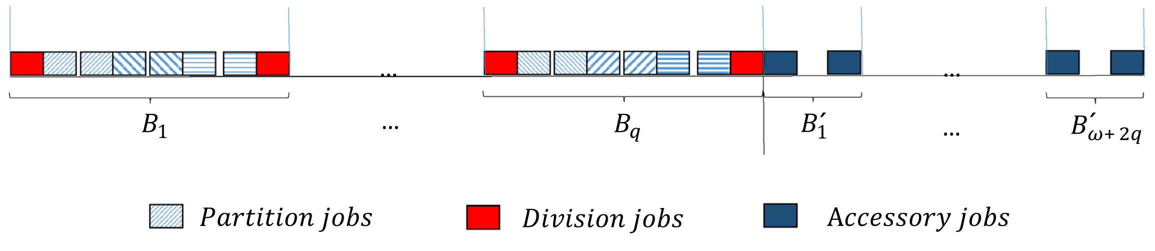

Fig. 2 A "yes" feasible schedule of instance $\mathcal{I}_1$ with two types of blocks

time lags. Therefore, the earliest possible starting time of the composite block B is $\beta + 1$, which implies that the total completion time of the ω medium jobs is at least $(\beta+1)\omega + (6E-1)\omega(\omega+1)/2 = \omega + \alpha$, i.e., $\sum C_j(\sigma) \geq y$.

Case 2: the density of at least one block in schedule σ is less than $1/(6E-1)$. All these low-density blocks must be scheduled after the composite block B in σ. Let us focus on the first such block B_k. Note that B_k must contain at least one large job, otherwise the density of the block would be at least $1/(6E-1)$. A time slot within the time lag of a large job is said to be *idle* if no operation is being processed during the slot and it is not part of the time lag of any small or medium job. Without changing the starting time of block B_k in schedule σ and the sequence of the job operations in the block, we move forward some operations, including the last one, in block B_k to eliminate an amount $\delta > 0$ (to be specified below) of idle time in block B_k and create a new block B_k', which is infeasible (due to smaller time lags between the two operations of some jobs), but has a smaller total job completion time.

Let λ be the length of block B_k, which contains $n_l \geq 1$ large jobs, $n_m \geq 0$ medium jobs and $n_s \geq 0$ small jobs. Denote by λ_s the total length of n_s small jobs and by λ_m the total length of n_m medium jobs. Let T be the total amount of idle time in B_k. Then, we have $\lambda = n_l(4E) + T + \lambda_m + \lambda_s$. Since the density of block B_k is smaller than $1/(6E-1)$, we have

$$\frac{n_l + n_m + n_s}{n_l(4E) + T + \lambda_m + \lambda_s} < \frac{1}{6E-1}, \quad \frac{n_m}{\lambda_m} = \frac{1}{6E-1}, \quad \frac{n_s}{\lambda_s} > \frac{1}{6E-1},$$

which imply $\frac{n_l}{n_l(4E)+T} < \frac{1}{6E-1}$ and hence $T > n_l(2E-1) \geq 2E-1 > 0$.

Let $\delta = T - n_l(2E-1)$ be the total amount of idle time to be eliminated from block B_k to create the new block B_k' defined earlier. Then B_k' has a density of at least $1/(6E-1)$ and still contains an amount $n_l(2E-1) > 0$ of idle time. On the other hand, the schedule σ_1, which is the same as σ except B_k is replaced by B_k', is infeasible but clearly has a smaller total job completion time than $\sigma : \sum C_j(\sigma_1) < \sum C_j(\sigma)$.

In the same way, by eliminating some idle time of each low-density block scheduled after the composite block B in schedule σ, we create an infeasible schedule (due to shorter time lags than required) σ_1, which satisfies $\sum C_j(\sigma_1) < \sum C_j(\sigma)$. In schedule σ_1, every block after the composite block B has a density of at least $1/(6E-1)$ and still contains some positive amount of idle time. Next, we move all these (infeasible) blocks before the composite block B to create another new (infeasible) schedule σ_2, which satisfies $\sum C_j(\sigma_2) \leq \sum C_j(\sigma_1)$.

Now in schedule σ_2, the first $6q$ jobs, which are scheduled before the last ω medium jobs, contain at least one positive amount of idle time, which implies that the total completion time of the first $6q$ jobs is strictly larger than β. Therefore, we obtain

$$\sum C_j(\sigma) > \sum C_j(\sigma_2) \geq \omega + \alpha = y,$$

which completes our proof of Case 2.

4.2 Problem $1|(a_j, L, b)|\sum C_j$

This subsection establishes the following result.

Theorem 2 *Problem $1|(a_j, L, b)|\sum C_j$ is strongly NP-hard.*

Given an instance of 3- Partition as specified in Sect. 3, we construct the following instance $\mathcal{I}_2$ of the decision version of problem $1|(a_j, L, b)|\sum C_j$. Let the common exact time delay $L = E$ and the common second-operation size $b = \min_{i=1,\ldots,3q} e_i$. There are $4q + \omega$ (with $\omega = 16q^2E$) jobs in three job sets: $3q$ small jobs in $\mathcal{J}_1$, which we call *partition* jobs, q identical medium jobs in $\mathcal{J}_2$, which we call *division* jobs, and ω identical large jobs in $\mathcal{J}_3$, which we call *accessory* jobs. More specifically,

$$\begin{aligned}
&\mathcal{J}_1 = \{J_j : a_j = e_j, j = 1, \ldots, 3q\}, && \text{(partition jobs)}\\
&\mathcal{J}_2 = \{J_j : a_j = E+1, j = 3q+1, \ldots, 4q\}, && \text{(division jobs)}\\
&\mathcal{J}_3 = \{J_j : a_j = 32q^2E^2, j = 4q+1, \ldots, 4q+\omega\}. && \text{(accessory jobs)}
\end{aligned}$$

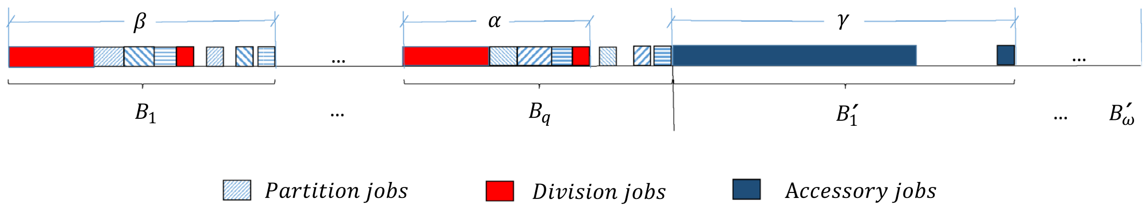

Fig. 3 A "yes" feasible schedule of instance $\mathcal{I}_2$ with two types of blocks

Is there a feasible schedule σ such that $\sum C_j(\sigma) < y$? Here,

$$y = q\alpha + (2q + 1 + \omega)\, q\beta + \frac{\omega(\omega+1)}{2}\gamma,$$

and $\alpha = 2E + b + 1$ and $\gamma = 32q^2E^2 + E + b$ are the lengths of a division and accessory job, respectively, while $\beta = 3E + b + 1$ is the length of a typical block of interest to be specified below.

First, suppose 3- PARTITION has a solution as specified in Sect. 3. Let us construct a "yes" feasible schedule as follows: it starts with q contiguous blocks, followed by ω contiguous accessory jobs as single-job blocks. Each of the first q blocks consists of one division job and three partition jobs of indices of I_k for $k = 1, \ldots, q$; the first operations of the three partition jobs are scheduled in the time lag of the division job, which implies that each of these blocks has a fixed length of β. We illustrate the whole schedule in Fig. 3.

Since each division job has a length of α, the completion time of the division job in block k is equal to $\alpha + (k-1)\beta$, which implies that the total completion time of the first q division jobs is $q\alpha + \frac{q(q-1)}{2}\beta$. Similarly, the total completion time of the last ω accessory jobs is $\omega(q\beta) + \frac{\omega(\omega+1)}{2}\gamma$, given γ is the length of a single accessory job. On the other hand, the total completion time of the three partition jobs in block k is less than $3k\beta (k = 1, \ldots, q)$. Hence, the total completion time of all partition jobs is less than $\frac{q(q+1)}{2}(3\beta)$. Therefore, the total job completion time $\sum C_j$ of the whole schedule is less than y.

Next, we show that, if there is an optimal schedule σ for problem $1|(a_j, L, b)|\sum C_j$ with $\sum C_j(\sigma) < y$, then 3-PARTITION must have a solution. First, we can easily verify the following:

Claim 1 *No two accessory and/or division jobs can be interleaved with each other, while an accessory job or a division job can be interleaved with any two partition jobs, but with at most three partition jobs.*

Claim 2 *According to Lemma 1, in any optimal schedule, any block of one division job and/or one or more partition jobs must be scheduled before any block of a single accessory job.*

Now let us verify the following claim:

Claim 3 *An accessory job is not interleaved with any partition job(s) in any optimal schedule.*

Given any schedule σ, assume an accessory job is interleaved with a single partition job. Let us break the block into two single-job blocks, starting (at the starting time of the original block) with partition job immediately followed by the accessory job. The partial schedule σ' (if any) after the original block is pushed backwards by $E + b$. Let us calculate the change of the total job completion time. The completion time of the partition job decreases by $32q^2E^2$, while that of the accessory job increases by less than $\frac{3E}{2} + b$. Since the completion time of each job in partial schedule σ' increases by $E + b$, the total completion time of these jobs increases by at most $(E + b)(4q + \omega - 2)$, which is less than $32q^2E^2 - (\frac{3E}{2} + b)$. Therefore, schedule σ cannot be optimal. Note that if an accessory job is interleaved with more than one partition job (actually at most three according to Claim 1 above), breaking the block into two and re-scheduling the block of partition job(s) just before the block of a single accessory job will decrease the total job completion time even more than in the previous case with one partition job.

As a result of Claims 1–3 above, all ω accessory jobs will be scheduled as single-job blocks at the end of any optimal schedule.

If 3- PARTITION has no solution, then $3q$ partition jobs and q division jobs cannot be processed to finish by time $q\beta$ (Orman and Potts 1997), which implies that the remaining ω accessory jobs will each be completed at least one time unit later than in the "yes" schedule we constructed earlier when 3- PARTITION had a solution. That is, the total completion time of these jobs will be at least $\omega = 16q^2E$ time units greater compared with $\omega(q\beta) + \frac{\omega(\omega+1)}{2}\gamma$. Together with $\omega > q\alpha + (2q + 1)\, q\beta$, this implies that $\sum C_j(\sigma) > y$.

4.3 Problem $1|(a, L, b_j)|\sum C_j$

This problem looks very similar to the one we considered in the preceding section. One might be tempted to use some kind of symmetry (by viewing the time dimension in an opposite direction) to claim strong NP-hardness of the current problem based on Theorem 2. Unfortunately, our objective function of $\sum C_j$, instead of $C_{\max}$, prevents a potential use of such sym-

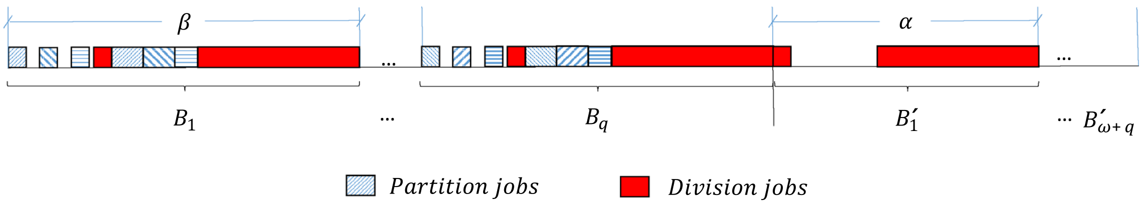

Fig. 4 A “yes” feasible schedule of instance $\mathcal{I}_3$ with two types of blocks

metry. Nonetheless, our current problem is indeed strongly NP-hard, as we show below.

Theorem 3 *Problem* $1|(a, L, b_j)|\sum C_j$ *is strongly NP-hard.*

Given an instance of 3- PARTITION as specified in Sect. 3, we construct the following instance $\mathcal{I}_3$ of the decision version of problem $1|(a, L, b_j)|\sum C_j$. Let the common first-operation size $a = \min_{i=1,\ldots,3q} e_i$ and the common exact time delay $L = E$. There are $5q + \omega$ jobs in two job sets: $3q$ small jobs in $\mathcal{J}_1$, which we call *partition* jobs, and $2q + \omega$ (with $\omega = 9q^2 A$ and $A = 12q^2 E$) identical large jobs in $\mathcal{J}_2$, which we call *division* jobs. More specifically,

$$\mathcal{J}_1 = \{J_j : b_j = e_j, j = 1, \ldots, 3q\}, \qquad \text{(partition jobs)}$$
$$\mathcal{J}_2 = \{J_j : b_j = A, j = 3q + 1, \ldots, 5q + \omega\}. \text{(division jobs)}$$

Is there a feasible schedule σ such that $\sum C_j(\sigma) < y$? Here,

$$y = 3q(2E + a) + (3q - 1 + \omega)\, q\beta + \frac{(q + \omega)(q + \omega + 1)}{2}\alpha,$$

and $\alpha = E + A + a$ is the length of each division job, while $\beta = 2E + A + a$ is the length of a typical block of interest to be specified below.

First, suppose 3- PARTITION has a solution as specified in Sect. 3. Let us construct a “yes” feasible schedule as follows: it starts with q contiguous blocks, followed by $q + \omega$ contiguous division jobs as single-job blocks. We illustrate the whole schedule in Fig. 4.

Each of the first q blocks consists of one division job and three partition jobs of indices of I_k for $k = 1, \ldots, q$; the second operations of the three partition jobs are scheduled in the time lag of the division job, which implies that each of these q blocks has a fixed length of β. Since each such block ends with a division job, the completion time of the division job in block k is equal to $k\beta$, which implies that the total completion time of the first q division jobs is $\frac{q(q+1)}{2}\beta$. Similarly, the total completion time of the last $q + \omega$ division jobs is $(q + \omega)(q\beta) + \frac{(q+\omega)(q+\omega+1)}{2}\alpha$, given that α is the length of a single division job. On the other hand, the total completion time of the three partition jobs in block k is less than $3(k\beta - A) = 3((k - 1)\beta + (2E + a))$ $(k = 1, \ldots, q)$. Hence, the total completion time of all partition jobs is less than $3q(2E + a) + \frac{3q(q-1)}{2}\beta$. Therefore, the total job completion time of the whole schedule is less than the sum of these three terms, which is equal to y.

Next, we show that, if there is an optimal schedule σ for problem $1|(a, L, b_j)|\sum C_j$ with $\sum C_j(\sigma) < y$, then 3-PARTITION must have a solution. We can easily verify the following, which imply that all $3q$ partition jobs are scheduled before at least ω division jobs as single-job blocks in σ:

Claim 4 *No two division jobs can be interleaved with each other, while a division job can be interleaved with any two partition jobs, but with at most three partition jobs. In any case, the division job always starts and finishes last in an interleaved block.*

Claim 5 *In any optimal schedule, if a block consists of only partition jobs, it must be scheduled before any block containing a division job.*

Claim 6 *When a division job is interleaved with one or more partition jobs, the interleaved block must be scheduled before any block of a single division job.*

Suppose 3- PARTITION has no solution; we are to derive a contradiction that $\sum C_j(\sigma) > y$. Let τ be the time when all $3q$ partition jobs and the first $2q$ division jobs are completed in σ. Let us consider two cases.

Case 1. $\tau > 2q\alpha + qE$. This implies that the last ω division jobs will each be completed at least one time unit later than in the “yes” schedule we constructed earlier when 3- PARTITION had a solution. That is, the total completion time of these jobs will be at least ω more compared with $\omega q(\alpha + \beta) + \frac{\omega(\omega+1)}{2}\alpha$. Note that

$$\frac{(q + \omega)(q + \omega + 1)}{2}\alpha = \omega q\alpha + \frac{q(q + 1)}{2}\alpha + \frac{\omega(\omega + 1)}{2}\alpha, \tag{1}$$

which together with

$$\omega = 9q^2 A > 3q(2E + a) + (3q - 1)q\beta + \frac{q(q + 1)}{2}\alpha$$

implies that $\sum C_j(\sigma) > y$.

Case 2. $\tau \leq 2q\alpha + qE$. According to the last part of Claim 4 above, interleaving any partition job $J_j \in \mathcal{J}_1$ into a block will increase the length of the block by at least e_j. Therefore, the total length of all blocks of the $3q$ partition jobs and the first $2q$ division jobs is at least $2q\alpha + qE \geq \tau$ if all partition jobs are interleaved with division jobs. This implies that in schedule σ there is no interleaved block consisting of only partition job(s), since such a block would *additionally* contribute to τ by at least E.

On the one hand, all partition jobs (in fact, at most three, according to Claim 4 above) interleaved with the kth division job have to be finished before the second operation of the division job starts (at a time no earlier than $(k-1)A$). On the other hand, at least one partition job has to be interleaved with a division job after the qth one, because 3- PARTITION has no solution. We conclude that the total completion time of all the $3q$ partition jobs is at least $A + 3\sum_{k=1}^{q}(k-1)A = A + \frac{3q(q-1)}{2}A$. As for the other jobs, the first $2q$ division jobs have a total completion time at least $\frac{2q(2q+1)}{2}\alpha$ and the remaining ω division jobs have a total completion time at least $\omega q(\alpha+\beta) + \frac{\omega(\omega+1)}{2}\alpha$. Consequently, with (1) we obtain a contradiction:

$$\begin{aligned}\sum C_j(\sigma) - y &\geq A + \frac{3q(q-1)}{2}A + \frac{q(3q+1)}{2}\alpha \\ &\quad - q(3q-1)\beta - 3q(2E+a) \\ &= A + \frac{q(3q+1)}{2}(E+a) \\ &\quad - q(3q+2)(2E+a) > 0.\end{aligned}$$

4.4 Problem $1|(p_j, p_j, p_j)|\sum C_j$

This subsection establishes the following result.

Theorem 4 *Problem* $1|(p_j, p_j, p_j)|\sum C_j$ *is strongly NP-hard.*

Given an instance of 3- PARTITION as specified in Sect. 3, we construct the following instance $\mathcal{I}_4$ of problem $1|(p_j, p_j, p_j)|\sum C_j$. There are $3q + v + \omega$ jobs in three job sets with $v = 8q$, $\omega = v + 2q + 120q^2$, and $\gamma = qE^2$:

$$\begin{aligned}\mathcal{J}_1 &= \{J_{1j} : a_{1j} = L_{1j} = b_{1j} = v(\gamma e_j + j), \\ &\qquad j = 1, \ldots, 3q\}, \qquad \text{(partition jobs)} \\ \mathcal{J}_2 &= \{J_{2j} : a_{2j} = L_{2j} = b_{2j} = 3v(\gamma E + 9q) + j, \\ &\qquad j = 1, \ldots, \omega\}, \qquad \text{(division jobs)} \\ \mathcal{J}_3 &= \{J_{3j} : a_{3j} = L_{3j} = b_{3j} = 3v(\gamma E + 9q)/4 + j, \\ &\qquad j = 1, \ldots, v\}. \qquad \text{(accessory jobs)}\end{aligned}$$

To facilitate our calculation, we introduce the following $3q$ *shadow* jobs:

$$\mathcal{J}_4 = \{J_{4j} : a_{4j} = L_{4j} = b_{4j} = 0,\ \ j = 1, \ldots, 3q\}.$$

Please note that shadow jobs should *not* be scheduled in any feasible schedule.

Denote by D the total job completion time of all ω division jobs in the following partial schedule σ_1: the jobs in $\mathcal{J}_2$ are processed as single-job blocks from time 0 according to sequence $(J_{21}, \ldots, J_{2\omega})$ without idle time between successive blocks.

Denote by S the total job completion time of all shadow jobs in the following *virtual* schedule σ_2': start with schedule σ_1, process all shadow jobs in the time lags of the first q division jobs, each time lag containing exactly three shadow jobs.

Denote by A the total job completion time of all accessory jobs in the following partial schedule σ_2: start with schedule σ_1, each accessory job J_{3j} is processed in the time lag of division job $J_{2,q+j}$ for $j = 1, \ldots, v$.

Let $\alpha = 3v(\gamma\frac{E}{2} + 3q)$, which is a (strict) upper bound on the length of any partition job. The decision version of problem $1|(p_j, p_j, p_j)|\sum C_j$ is: does there exist a schedule σ of all jobs such that $\sum C_j(\sigma) < y$, where $y = D + S + A + 6q\alpha$?

Suppose 3- PARTITION has a solution as specified in Sect. 3. We first construct schedule σ_2 as above. Then, we construct schedule σ of all jobs by adding all partition jobs into σ_2 according to the 3-job groups $\{I_k : k = 1, \ldots, q\}$, processing the three jobs $\{J_{1j} : j \in I_k\}$ without interleaving in the time lag of division job J_{2k} for $k = 1, \ldots, q$. We illustrate the whole schedule in Fig. 5.

Consider a 3-job group of partition jobs in groups $\{I_k : k = 1, \ldots, q\}$. If the group of three jobs are scheduled without interleaving from time 0, then the total job completion time of the group is less than 6α, where α is known to be an upper bound of the length of any partition job. The total job completion time of all partition jobs in schedule σ equals the total job completion time of all shadow jobs in virtual schedule σ_2' plus that of q 3-job groups each scheduled from time 0, which is therefore less than $S + 6q\alpha$. Therefore, $\sum C_j(\sigma) < D + A + (S + 6q\alpha) = y$.

Now we show that if there is an optimal schedule σ such that $\sum C_j(\sigma) < y$, then 3- PARTITION has a solution.

Given that the operations of each (non-shadow) job have a different size from those of any other job, if two jobs are interleaved, then both operations of one of these two jobs are processed entirely within the time lag of the other job. Apart from the shadow jobs, the partition jobs have the smallest operations, while the division jobs have the biggest operations. According to the sizes of job operations, we can easily establish the following claims in any feasible schedule:

Claim 7 *At least* 2 *partition jobs or* 1 *accessory job can be scheduled within the time lag of any division job. On the other hand, at most* 3 *partition jobs or* 1 *accessory job can be scheduled within the time lag of any division job.*

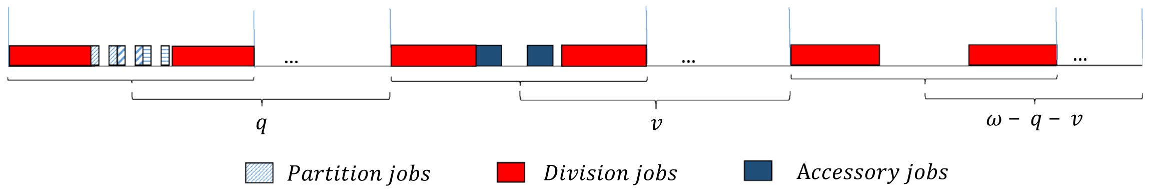

Fig. 5 A "yes" feasible schedule of instance $\mathcal{I}_4$ with three types of blocks

Claim 8 *No partition job can be scheduled within the time lag of any accessory job.*

Claim 9 *If 3 partition jobs $\{J_{1i_1}, J_{1i_2}, J_{1i_3}\}$ are scheduled within the time lag of a division job, then we must have $e_{i_1} + e_{i_2} + e_{i_3} \le E$.*

If a partition job or an accessory job forms a single-job block, then such a job is said to be *independent*. From Claims 7 and 8 above, all $3q$ partition jobs can be scheduled within the time lags of $2q$ division jobs and all v accessory jobs can be scheduled within the time lags of v division jobs. Therefore, according to Lemma 1 in Sect. 3, no partition job or accessory job should be scheduled after the first $2q + v$ division jobs in optimal schedule σ. We claim that there is no independent job in schedule σ. In fact, if this is not the case, then we reschedule the last independent job to a time lag of the first $2q + v$ division jobs, each having a length at most $\beta_1 := 9v(\gamma E + 9q) + 3(2q + v)$, which will increase the completion time of this job by at most $(2q + v)\beta_1$ and, at the same time, decrease the completion time of each of the last $\omega - (2q + v)$ division jobs by at least $3v\left(\gamma \frac{E}{4} + 1\right)$, a lower bound of the length of any partition job, while the completion time of any other job cannot increase. Since $(\omega - (2q + v))\left(3v\left(\gamma \frac{E}{4} + 1\right)\right) > (2q + v)\beta_1$, the optimality of schedule σ is contradicted.

Therefore, in σ all partition and accessory jobs are scheduled within time lags of division jobs, which enables us to easily verify based on Lemma 1 that all accessory jobs are scheduled after any partition job, and all accessory jobs and division jobs are scheduled in the order of increasing sizes of their operations in schedule σ.

We further claim that all partition jobs in σ occupy the time lags of at most q division jobs, which implies that the time lag of each of these q division job accommodates exactly 3-partition jobs, which (together with Claim 9 above) in turn implies that 3- PARTITION has a solution. Suppose to the contrary that our claim is false and the partition jobs occupy the time lags of at least $q + 1$ division jobs in schedule σ, which together with Claim 7 listed above implies that, in comparison with partial schedule σ_2 constructed earlier, each accessory job in σ is delayed by at least $\beta_0 = 9v(\gamma E + 9q) + 3$, which is a lower bound on the length of any division job, and hence, the total completion time of all accessory jobs in σ is increased by at least $v\beta_0$. Since $v\beta_0 > 6q\alpha$ and noticing that the total completion time of all partition jobs in σ is bounded below by S, we obtain $\sum C_j(\sigma) \ge D + (A + v\beta_0) + S > y$, which is in contradiction to our assumption that $\sum C_j(\sigma) < y$.

5 Polynomially solvable problems

In this section, we establish that, apart from those problems identified in the preceding section as strongly NP-hard problems, all remaining problems are polynomially solvable. To this end, we provide a polynomial-time algorithm for a problem in each subsection. Let the job set be $\mathcal{J} = \{1, \ldots, n\}$.

5.1 Problem $1|(p, p, b_j)|\sum C_j$

Fix any optimal schedule σ and let us derive some properties for σ. First, we observe that any block of σ consists of at most two jobs. We have the following immediate observation.

Observation 1 *If a job can be interleaved with another job, then the two jobs as a block have a smaller total job completion time than processing them one after another as two single-job blocks.*

We call the two jobs an *interleaved pair*, denoted by (j, k), and job j the *head* of the pair and job k the *tail* of the pair. The following lemma is then immediate:

Lemma 2 *Job j, as the head of an interleaved pair, has $b_j \le p$. The heads of any two interleaved pairs can be swapped without changing the total job completion time.* □

The following lemma describes a basic structure of schedule σ, which is further refined in Lemma 4 according to Lemma 1.

Lemma 3 *In optimal schedule σ all single-job blocks are scheduled after all interleaved pairs.*

Proof Suppose to the contrary that job j is scheduled just before an interleaved pair (j', k) in σ. Then, the makespan of the partial schedule of these three jobs is $5p + b_j + b_k$. Let us *improve* the partial schedule while keeping its makespan unchanged, which directly contradicts the optimality of σ. In fact, if we form a new interleaved pair (j', j) followed by job k, then the total completion time is decreased by $p + b_j$. □

Partition the job set $\mathcal{J}$ into two subsets $S = \{j : b_j \leq p\}$ and $B = \{j : b_j > p\}$. According to Observation 1 and Lemmas 2 and 3, we have the following description for the structure of optimal schedules.

Lemma 4 *Optimal schedule σ consists of a sequence of blocks of the following form:*

$$\left((j'_1, j_1), \ldots, (j'_k, j_k), j_{k+1}, \ldots, j_{k+\ell}\right),$$

where $k, \ell \geq 0$ are two integers, $\{j'_1, \ldots, j'_k\} \subseteq S$, and $b_{j_1} \leq b_{j_2} \leq \cdots \leq b_{j_k} \leq b_{j_{k+1}} \leq \cdots \leq b_{j_{k+\ell}}$. Furthermore, if $j_{k+1} \in S$, then $\ell = 1$ and $B = \emptyset$; otherwise, there is $0 \leq s \leq k$ such that $b_{j_s} \leq \min\{b_{j'_1}, \ldots, b_{j'_k}\}$, and $\{j_1, \ldots, j_s\} \cup \{j'_1, \ldots, j'_k\} = S$ and $\{j_{s+1}, \ldots, j_{k+\ell}\} = B$. □

If $B = \emptyset$, an optimal schedule can be found easily by maximizing the number of interleaving pairs. If $B \neq \emptyset$, according to Lemma 4, we need to find the integer value s. Note that s must satisfy $\max\{0, \lfloor(|S| - |B|)/2\rfloor\} \leq s \leq \lfloor|S|/2\rfloor$, and hence, s can be found easily by enumeration. Formally, the following algorithm finds an optimal schedule.

Algorithm 1

Step 1: Reindex all jobs if necessary so that $b_1 \leq b_2 \leq \cdots \leq b_n$. Partition the job set $\mathcal{J}$ into two subsets $S = \{j : b_j \leq p\}$ and $B = \{j : b_j > p\}$. Let $s_1 = \max\{0, \lfloor(|S| - |B|)/2\rfloor\}$, $s'_2 = \lfloor n/2\rfloor$ and $s''_2 = \lceil n/2\rceil$.
Step 2: If $B = \emptyset$, then set $s = s''_2$, $H = \{s+1, \ldots, n\}$ and $T = \{1, \ldots, s\}$; perform Step 3 and stop with optimal schedule $\sigma(s)$. Otherwise ($B \neq \emptyset$), iteratively for $s = s'_2, \ldots, s_1$, set $H = \{s+1, \ldots, |S|\}$ and $T = \{1, \ldots, s\} \cup B$ and perform Step 3, and then stop with optimal schedule $\sigma(s^*)$, where $s^* = \arg\min\{X(s) : s = s_1, \ldots, s'_2\}$.
Step 3 (subroutine): (a) Interleave each job in H with each of the first $|H|$ jobs in T and schedule the $|H|$ interleaved pairs in order of increasing indices in T, and then append the remaining $|T| - |H|$ jobs in T to the end of the schedule in order of increasing indices in T; (b) record the resulting full schedule as $\sigma(s)$ and the corresponding total completion time as $X(s)$.

Theorem 5 *Algorithm 1 generates an optimal schedule in $O(n \log n + n^2)$ time for problem $1|(p, p, b_j)|\sum C_j$.* □

5.2 Problem $1|(a_j, p, p)|\sum C_j$

This problem is seemingly symmetric to the one we have dealt with in the preceding section. However, as we point out at the beginning of Sect. 4.3, such a symmetry does not hold due to the asymmetric nature of our objective function and hence Algorithm 1 cannot be directly applied here. Nonetheless, the algorithm can be slightly modified for our new problem.

First, Observation 1 still holds here. With the same notion of an interleaved pair there, we have the following lemma, symmetric to Lemma 2.

Lemma 5 *Job k, as the tail of an interleaved pair, has $a_k \leq p$. The tails of any two interleaved pairs can be swapped without changing the total job completion time.* □

As before, we fix any optimal schedule σ and derive some properties for σ. It can be verified that Lemma 3 still holds here. Now partition the job set $\mathcal{J}$ into two subsets $S' = \{j : a_j \leq p\}$ and $B' = \{j : a_j > p\}$. According to Observation 1 and Lemmas 5 and 3, we have the following description for the structure of optimal schedules.

Lemma 6 *Optimal schedule σ consists of a sequence of blocks of the following form:*

$$\left((j_1, j'_1), \ldots, (j_k, j'_k), j_{k+1}, \ldots, j_{k+\ell}\right),$$

where $k, \ell \geq 0$ are two integers, $\{j'_1, \ldots, j'_k\} \subseteq S'$, and $a_{j_1} \leq a_{j_2} \leq \cdots \leq a_{j_k} \leq a_{j_{k+1}} \leq \cdots \leq a_{j_{k+\ell}}$. Furthermore, if $j_{k+1} \in S'$, then $\ell = 1$ and $B' = \emptyset$; otherwise, there is $0 \leq s \leq k$ such that $a_{j_s} \leq \min\{a_{j'_1}, \ldots, a_{j'_k}\}$, and $\{j_1, \ldots, j_s\} \cup \{j'_1, \ldots, j'_k\} = S'$ and $\{j_{s+1}, \ldots, j_{k+\ell}\} = B'$. □

Now we can provide a modified version of Algorithm 1 to find an optimal schedule for problem $1|(a_j, p, p)|\sum C_j$ as follows.

Algorithm 2

Step 1: Reindex all jobs if necessary so that $a_1 \leq a_2 \leq \cdots \leq a_n$. Partition the job set $\mathcal{J}$ into two subsets $S' = \{j : a_j \leq p\}$ and $B' = \{j : a_j > p\}$. Let $s_1 = \max\{0, \lfloor(|S'| - |B'|)/2\rfloor\}$, $s'_2 = \lfloor n/2\rfloor$ and $s''_2 = \lceil n/2\rceil$.
Step 2: If $B' = \emptyset$, then set $s = s''_2$, $T = \{s+1, \ldots, n\}$ and $H = \{1, \ldots, s\}$; perform Step 3 and stop with optimal schedule $\sigma(s)$. Otherwise ($B' \neq \emptyset$), iteratively for $s = s'_2, \ldots, s_1$, set $T = \{s+1, \ldots, |S'|\}$ and $H = \{1, \ldots, s\} \cup B'$ and perform Step 3, and then stop with optimal schedule $\sigma(s^*)$, where $s^* = \arg\min\{X(s) : s = s_1, \ldots, s'_2\}$.
Step 3 (subroutine): (a) Interleave each job in T with each of the first $|T|$ jobs in H and schedule the $|T|$ interleaved pairs in order of increasing indices in H, and then append the remaining $|H| - |T|$ jobs in H to the end of the schedule in order of increasing indices in H; (b) record the resulting full schedule as $\sigma(s)$ and the corresponding total completion time as $X(s)$.

Theorem 6 *Algorithm 2 generates an optimal schedule in $O(n \log n + n^2)$ time for problem $1|(a_j, p, p)|\sum C_j$.* □

5.3 Problem $1|(p, L, p)|\sum C_j$

In this problem, all operations are of equal length. The algorithm developed by Orman and Potts (1997) for problem $1|(p, L, p)|C_{\max}$ can be adapted here to find an optimal solution to this problem. For the sake of completeness, we restate their algorithm below.

Algorithm 3 Initialization: Let $\eta = \lfloor L/p \rfloor$ and $k = \lfloor n/(\eta + 1) \rfloor$.

```
Step 1: Form $k$ blocks of jobs, each consisting of $\eta + 1$ jobs with their first operations processed contiguously. Schedule all $k$ blocks contiguously.
Step 2: Immediately after the $k$ blocks, process the remaining jobs (if any) with their first operations scheduled contiguously.
```

Using similar arguments to those in Orman and Potts (1997), one can easily verify that Algorithm 3 identifies an optimal schedule. Hence, we have the following theorem.

Theorem 7 *Algorithm 3 generates an optimal schedule for problem $1|(p, L, p)|\sum C_j$ in $O(n)$ time.* □

5.4 Problem $1|(a, L, b)|\sum C_j$

This problem is more general than the one considered in the preceding section. However, we make it a bit easier by assuming that the integer parameters a, b and L are fixed. We assume that $\max\{a, b\} \le L < (n-1)\max\{a, b\}$. Otherwise, the problem becomes trivial: the schedule of either n single-job blocks or one n-job block is optimal.

As can be seen from the solution approach below, we can assume $a \ge b$ without loss of generality. We see that all jobs are identical, and for convenience we index the jobs according to their sequence of being processed. Let us apply the idea of *pattern* used in Ahr et al. (2004) and Baptiste (2010) for our analysis. For any partial schedule of jobs $\{1, \ldots, k\}$ $(1 \le k \le n)$, the *pattern* $p = p(a, L, b) = (p[1], \ldots, p[L])$ of the partial schedule is defined as the L-element sequence of 0 and 1, which indicates that the machine is idle or busy, respectively, during the L time slots of the time lag of job k. That is, we use patterns to describe the status of machine during the time lag of the last job in the partial schedule. It is easy to see that, in any such pattern, all 1's are in blocks of length b and each such block is followed by at least $(a - b)$ 0's. Let $P(a, L, b)$ be the set of all patterns $p = p\,(a, L, b)$. The total number of patterns is bounded: $|P(a, L, b)| \le \gamma \equiv a^{\frac{L}{a-1}}$ for $a > 1$ (Ahr et al. 2004).

When a job is added to a partial schedule, the pattern of the original partial schedule becomes the pattern of the new partial schedule. The scheduling decision for scheduling the additional job is then equivalent to a pattern selection. A partial schedule of one job has a pattern $p_0 = (0, \ldots, 0)$ with the entire time lag idle. When we append (not interleave) a job to a partial schedule of pattern p, the pattern of the new partial schedule becomes p_0, indicating that the makespan increases by $a + L + b$, which we define as the *distance* from pattern p to pattern p_0. On the other hand, if the additional job is interleaved into the existing partial schedule, resulting in a new pattern q, so that its first operation is processed from slot $\lambda - a + 1$ to slot λ of the time lag of the last job ($\lambda \ge a$), then

$$\begin{cases} q[i] = p[i + \lambda], & i = 1, \ldots, L - \lambda; \\ p[\lambda - a + i] = 0, & i = 1, \ldots, a. \end{cases} \tag{2}$$

Consequently, the makespan of the new partial schedule is that of the original partial schedule plus λ. Therefore, we define the *distance* from pattern p to pattern q ($q \ne p_0$) as $\Delta_{pq} = \min \lambda$, where λ satisfies (2) and $\Delta_{pq} = +\infty$ if no λ can satisfy (2). Hence, distance from one pattern to another can be calculated in $O(L^2)$ time.

Any given schedule corresponds to a unique path with n vertices of patterns and any path with n vertices of patterns represents a unique schedule. If partial schedule $\{1, \ldots, k\}$ has pattern p and addition of job $k + 1$ results in pattern q, then we have $C_{k+1} = C_k + \Delta_{pq}$, where C_k and C_{k+1} are completion time of job k and $k + 1$, respectively. Initially, we have $C_1 = a + L + b$. The total job completion time of a schedule can therefore be calculated by accumulating distances between consecutive patterns.

Next, we will show that all paths with n vertices of patterns can be fully enumerated in polynomial time when a, b, L are fixed. Similar to Baptiste (2010), we define an elementary path as a sequence of different vertices and an elementary cycle as $[x, \sigma, x]$ where x is a vertex and σ is an elementary path. A dominant path is a path of n vertices of patterns in which all identical elementary cycles (if any) are consecutive.

Lemma 7 *For problem $1|(a, L, b)|\sum C_j$, there exists an optimal path of patterns that is a dominant one.*

Proof Suppose there is an optimal path that contains an identical elementary cycle $[x, \sigma, x]$, which appears more than once in the path but not consecutive. More specifically, let $[\pi_1, x, \sigma, x, \pi_2, x, \sigma, x, \pi_3]$ be an optimal path, where π_1, π_2 and π_3 are some paths with $|\pi_2| > 0$. Then it is easy to verify that either $[\pi_1, x, \pi_2, x, \sigma, x, \sigma, x, \pi_3]$ or $[\pi_1, x, \sigma, x, \sigma, x, \pi_2, x, \pi_3]$ will decrease the total job completion time (which is impossible due to optimality of the original path), unless all three paths of n vertices of patterns,

$[\pi_1, x, \pi_2, x, \sigma, x, \sigma, x, \pi_3]$, $[\pi_1, x, \sigma, x, \sigma, x, \pi_2, x, \pi_3]$ and $[\pi_1, x, \sigma, x, \pi_2, x, \sigma, x, \pi_3]$, have the same total job completion time. We have therefore constructed an optimal path that is dominant. □

The following two lemmas are proved by Baptiste (2010). We list them here for the sake of completeness.

Lemma 8 (Baptiste 2010) *An optimal and dominant path has the following structure:*

$$\left[\pi_1, (\sigma_1)^{q_1}, \pi_2, (\sigma_2)^{q_2}, \ldots, \pi_h, (\sigma_h)^{q_h}, \pi_{h+1}\right], \qquad (3)$$

where (a) $h \le v = \gamma^\gamma$, (b) π_j $(j = 1, \ldots, h+1)$ and σ_i $(i = 1, \ldots, h)$ are elementary paths and the last vertex of π_i is also the last vertex of σ_i $(i = 1, \ldots, h)$, (c) q_i is a nonnegative integer value, which stands for the repeating times of σ_i $(i = 1, \ldots, h)$ in the optimal dominant path, and (d) $\sum_{i=1}^{h+1} |\pi_i| + \sum_{i=1}^{h} q_i|\sigma_i| = n$.

Let us call $[\pi_1, \sigma_1, \pi_2, \sigma_2, \ldots, \pi_h, \sigma_h, \pi_{h+1}]$ the *backbone* of the path defined by (3).

Lemma 9 (Baptiste 2010) *There are at most v^{2v+3} distinct backbones, where $v = \gamma^\gamma$.*

Lemma 10 *Given a backbone $[\pi_1, \sigma_1, \ldots, \pi_h, \sigma_h, \pi_{h+1}]$, the optimal value of $q_1, \ldots, q_h$ for $[\pi_1, (\sigma_1)^{q_1}, \pi_2, (\sigma_2)^{q_2}, \ldots, \pi_h, (\sigma_h)^{q_h}, \pi_{h+1}]$ can be found in $O(n^h)$ time.* □

Because h and the total number of backbones are bounded by fixed values when a, b, L are fixed, we have the following theorem.

Theorem 8 *An optimal schedule of problem $1|(a, L, b)|\sum C_j$ can be generated in $O(n^h)$ time, where h is a fixed value given that a, b, L are fixed.* □

6 Concluding remarks

In this paper, we have drawn a full complexity picture for single-machine scheduling of coupled tasks of exact time delays in between, with the objective of minimizing the total of job completion times. We hope our research will serve as a first step to filling in the blanks on research of the basic single-machine scheduling of coupled tasks with an objective function other than the makespan.

Acknowledgements This work is supported to the first author in part by a Visiting Professorship of the School of Management, Fudan University, and to the second author in part by the National Natural Science Foundation of China (Grant No. 71971065 and No. 71531005).

References

Ahr, D., Bekesi, J., Galambos, G., Oswald, M., & Reinelt, G. (2004). An exact algorithm for scheduling identical coupled tasks. *Mathematical Methods of Operations Research*, *59*, 193–203.

Baptiste, P. (2010). A note on scheduling identical coupled tasks in logarithmic time. *Discrete Applied Mathematics*, *158*, 583–587.

Bessy, S., & Giroudeau, R. (2019). Parameterized complexity of a coupled-task scheduling problem. *Journal of Scheduling*, *22*, 305–313.

Blazewicz, J., Pawlak, G., Tanas, M., & Wojciechowicz, W. (2012). New algorithms for coupled tasks scheduling: a survey. *RAIRO Operations Research*, *46*, 335–353.

Condotta, A., & Shakhlevich, N. V. (2012). Scheduling coupled-operation jobs with exact time-lags. *Discrete Applied Mathematics*, *160*, 2370–2388.

Elshafei, M., Sherali, H. D., & Smith, J. C. (2004). Radar pulse interleaving for multitarget tracking. *Naval Research Logistics*, *51*(1), 72–94.

Farina, A., & Neri, P. (1980). Multitarget interleaved tracking for phased-array radar. *IEE Proceedings F (Communications, Radar and Signal Processing)*, *127*(4), 312–318.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Operations Research*, *5*, 287–326.

Huo, Y., Li, H., & Zhao, H. (2009). Minimizing total completion time in two-machine flow shops with exact delays. *Computers & Operations Research*, *36*(6), 2018–2030.

Hwang, F. J., & Lin, B. M. T. (2011). Coupled-task scheduling on a single machine subject to a fixed-job-sequence. *Computers & Industrial Engineering*, *60*(4), 690–698.

Izquierdo-Fuente, A., & Casar-Corredera, J. R. (1994). Optimal radar pulse scheduling using a neural network. *IEEE International Conference on Neural Networks*, *7*, 4588–4591.

Khatami, M., Salehipour, A., & Cheng, T. C. E. (2020). Coupled task scheduling with exact delays: Literature review and models. *European Journal of Operational Research*, *282*(1), 19–39.

Leung, J. Y. T., Li, H., & Zhao, H. (2007). Scheduling two-machine flow shops with exact delays. *International Journal of Foundations of Computer Science*, *18*(02), 341–359.

Orman, A. J., Potts, C. N., Shahani, A. K., & Moore, A. R. (1996). Scheduling for a multifunction phased array radar system. *European Journal of Operational Research*, *90*(1), 13–25.

Orman, A. J., & Potts, C. N. (1997). On the complexity of coupled-task scheduling. *Discrete Applied Mathematics*, *72*, 141–154.

Orman, A. J., Shahani, A., & Moore, A. (1998). Modelling for the control of a complex radar system. *Computers & Operations Research*, *25*(3), 239–249.

Shapiro, R. D. (1980). Scheduling copuled tasks. *Naval Research Logistics Quarterly*, *28*, 489–497.

Simonin, G., Darties, B., Giroudeau, R., & König, J. C. (2011). Isomorphic coupled-task scheduling problem with compatibility constraints on a single processor. *Journal of Scheduling*, *14*, 501–509.

Yu, W., Hoogeveen, H., & Lenstra, J. K. (2004). Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard. *Journal of Scheduling*, *7*, 333–348.

Zhang, H., Xie, J., Ge, J., Zhang, Z., & Zong, B. (2019a). A hybrid adaptively genetic algorithm for task scheduling problem in the phased array radar. *European Journal of Operationsl Research*, *272*, 868–878.

Zhang, H., Xie, J., Ge, J., Shi, J., & Lu, W. (2019b). Optimization model and online task interleaving scheduling algorithm for MINO radar. *Computers & Industrial Engineering*, *127*, 865–874.