Coupled task scheduling with exact delays: Literature review and models

Mostafa Khatami [*] Amir Salehipour [†] T.C.E. Cheng [‡]

Abstract

The coupled task scheduling problem concerns scheduling a set of jobs, each with at least two tasks and there is an exact delay period between two consecutive tasks, on a set of machines to optimize a performance criterion. While research on the problem dates back to the 1980s, interests in the computational complexity of variants of the problem and solution methodologies have been evolving in the past few years. This motivates us to present an up-to-date and comprehensive literature review on the topic. Aiming to provide a complete road map for future research on the coupled task scheduling problem, we discuss all the relevant studies and potential research opportunities. In addition, we propose several sets of benchmark instances for the problem in various settings and provide a detailed evaluation of all the available models with a view to facilitating future research on the solution methods.

Key words: scheduling; coupled task; review; single machine; shop setting; benchmark instances; mathematical model

1 Introduction

In the coupled task scheduling problem, every job consists of at least two separated tasks, whereby the succeeding task must be processed after the completion of the first (preceding) task and there is an exact time interval between the tasks. Shapiro (1980) introduced the coupled task scheduling problem to model a pulsed radar system, where a pulse of electromagnetic energy is used to track an object. The pulse is transmitted and its reflection is received after a period of time to measure the size and/or shape of the tracked object. The concept of exact delays between consecutive tasks of a job has been applied in scheduling problems in the shop setting as well, e.g., in both the flow shop (Leung et al., 2007) and open shop (Ageev, 2018) environments.

To the best of our knowledge, Blazewicz et al. (2012) provided the only review of research on coupled task scheduling. Their review includes (1) discussing known complexity results for the problem, (2) reviewing the state-of-the-art algorithms, and (3) evaluating the performance of algorithms over a set of instances with up to 838 tasks. We present an updated and comprehensive review of coupled task scheduling with a view to (1) reviewing both the single-machine and shop settings, (2) generating sets of benchmark instances for the problem, and (3) discussing all the available models for the problem and evaluating their performance over the sets of generated instances.

In total, we review 39 published papers on the coupled task scheduling problem. The first study on the problem appeared in 1980. Between 1980 and 2006, only a few papers on the problem were published. Interestingly, almost 72% of the related studies were published after 2006. This is an indication of the current research trend for the problem. Figure 1 shows the number of papers published over the years. In terms of the scheduling environment, research on the coupled task scheduling problem can be divided into two categories, namely the single-machine and shop settings. There are 26 publications that study the problem in the single-machine setting, and ten papers studying the problem in the shop scheduling setting. Three papers study the problem in both settings. Out of the 13 publications that study the problem in the shop setting, 12 deal with flow shop scheduling and one paper considers the open shop environment. Table 1 summaries the outlets that publish the available studies on the coupled task scheduling problem (the entries are sorted in non-increasing order of the number of published papers). As the table shows, the majority of the papers are published in conference proceedings and the Journal of Scheduling.

1.1 Application

Shapiro (1980) presented the first application of the coupled task scheduling problem as follows: In a radar tracking system, pulses are transmitted and reflections are received once every specified update period. The radar cannot transmit a pulse at the same time when a reflected pulse is arriving; also, two reflected pulses cannot overlap. The radar devices transmit certain pulses to calculate the sizes, shapes, and speeds of the tracked objects. Therefore, one can model the transmission and reflection of pulses as two tasks with a fixed duration of delay between them. It is important that the idle time of the radar system is minimized. Farina and Neri (1980) studied multi-target tracking with the objective of minimizing the total radar time. Izquierdo-Fuente and Casar-Corredera (1994) focused on maximizing the number of targets that a multi-function radar can handle. Orman et al. (1996) studied a real-time scheduling problem for a

[*]School of Mathematical and Physical Sciences, University of Technology Sydney, Australia. Email: mostafa.khatami@student.uts.edu.au
[†]Corresponding author. School of Mathematical and Physical Sciences, University of Technology Sydney, Australia. Email: amir.salehipour@uts.edu.au
[‡]PolyU Business School, The Hong Kong Polytechnic University, Hong Kong. Email: edwin.cheng@polyu.edu.hk

Table 1: Distribution of papers by journals and conference proceedings.

Journal/Proceeding	Number of papers (%)
Proceedings	7 (17.95)
Journal of Scheduling	5 (12.82)
Computers & Operations Research	3 (7.69)
Discrete Applied Mathematics	3 (7.69)
RAIRO Operations Research	3 (7.69)
Computers & Industrial Engineering	2 (5.13)
International Journal of Production Research	2 (5.13)
Mathematical Methods of Operations Research	2 (5.13)
Naval Research Logistics	2 (5.13)
Operations Research Letters	2 (5.13)
Annals of Operations Research	1 (2.56)
Discrete Optimization	1 (2.56)
European Journal of Industrial Engineering	1 (2.56)
European Journal of Operational Research	1 (2.56)
International Journal of Foundations of Computer Science	1 (2.56)
International Journal of Operational Research	1 (2.56)
International Journal of Planning and Scheduling	1 (2.56)
The Journal of the Operational Research Society	1 (2.56)

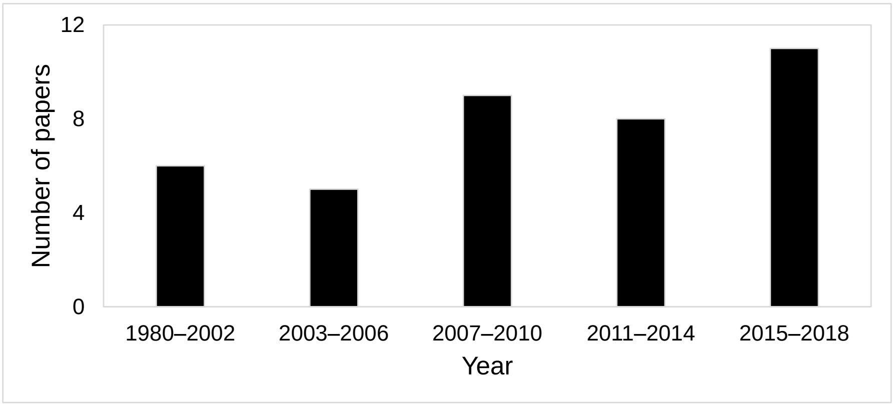

Figure 1: Number of papers published in different time periods.

multi-function-phase array radar system, where a job consists of coupled tasks. A multi-function radar system involves a number of dedicated radars that form a single system. Orman et al. (1998) also provided a model for complex radar system management. Elshafei et al. (2004) proposed integer programming models for the multi-target tracking system. They considered two objective functions, namely minimizing the total required time to scan all the targets (equivalent to the makespan), and minimizing the total risk of the unscanned targets within a time limit (equivalent to scanning as many targets with higher risks as possible in a given time limit), where there is a risk (cost) for a target remains unscanned. In a similar context, Simonin (2009) made use of the coupled task scheduling problem to improve the performance of submarine torpedoes. Various environmental data must be processed by sensors located on the torpedo. Here, the initial task is the transmission of a pulse from the torpedo to the water and the completion task consists of receiving the echo, while there is an exact duration of idle time between the two tasks.

In the flow shop setting, Ageev and Baburin (2007) modelled a chemistry manufacturing process as the coupled task scheduling problem. Specifically, two different operators execute the operations successively, where there is an exact technological delay between the finishing time of the first task and the starting time of the second one. Brauner et al. (2009) discussed the link between the coupled task and the single-machine no-wait robotic cell problem. In particular, a workstation in a cellular manufacturing system consists of an input station, a machine, and an output station. Such an environment can be modelled as the single-machine coupled task problem.

Condotta and Shakhlevich (2014) modelled a healthcare problem in the context of scheduling patient appointments in a chemotherapy outpatient clinic as the coupled task scheduling problem. The defining characteristic is the time lag that must elapse between every two appointments of the same patient.

1.2 Scope and classification

This review covers all the studies on scheduling problems with the coupled task characteristic, i.e., existence of fixed delays between consecutive tasks of a job. Specifically, each job may consist of at least two tasks, where there is a fixed amount of delay between every two consecutive tasks. The fixed delay therefore represents a strict lag time between the completion time of the preceding task and the starting time of the succeeding task. In the single-machine setting, a job includes two tasks, while in the shop setting, each job consists of at least two tasks. There are a number of related problems in the literature that we do not cover in this review.

There is research on scheduling problems with flexible delays, in which a lower and/or an upper bound on the duration of the delay is/are given. The lower bound implies that the delay between two consecutive tasks of a job must not be smaller than the minimum delay and the upper bound means the delay cannot be greater than the given bound. We can regard the coupled task scheduling problem as a special case of scheduling with flexible delays, where the minimum and maximum delays for each job are equal (see Figure 2). We refer the interested reader to Dell'Amico (1996); Potts and Whitehead (2007); Zhang and Van De Velde (2010) for reviews of research on scheduling problems with flexible delays.

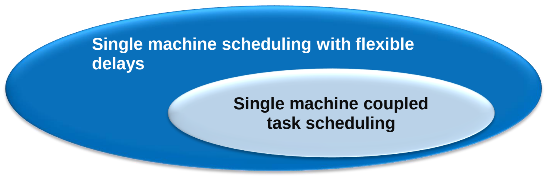
Single machine scheduling with flexible delays
Single machine coupled task scheduling

Figure 2: Single-machine with flexible delays and single-machine coupled task scheduling.

The coupled task scheduling problem in the shop environment is related to the well-known "no-wait" condition in shop scheduling. The no-wait assumption implies that the jobs are processed on all the machines without any delay between them. Therefore, we can regard the no-wait shop scheduling problem as a special case of the coupled task scheduling problem, where the value of the delay between every pair of tasks is equal to zero (see Figure 3). Allahverdi (2016) discussed research on no-wait shop scheduling.

The motivation for incorporating delays and modelling the feature as the coupled task scheduling problem stems from two reasons. First, it can be used to model a no-wait problem, where there is a transport time (or any processing that does not require a machine) between two consecutive tasks of the same job. That time requirement can indeed be represented by the delay duration in the coupled task scheduling problem. Chu and Proth (1996) presented a class of problems in which a transport time between successive tasks of a job occurs. Fondrevelle et al. (2009) discussed similar situations arising in the manufacturing process of thermic paper that involves chemical processing, where temporal constraints are imposed on the process. Likewise, such constraints must be satisfied in every processing step of a pharmaceutical plant, from raw materials and resource preparation to packaging.

Second, it can be used to model the no-wait condition involving bottleneck machines. According to Mitten (1959), the two-machine flow shop problem with delays can present the industrial environment involving two bottleneck machines, where the jobs must undergo a number of tasks on some intermediate machines between the two bottleneck machines. The elapsed time on those intermediate machines can be represented by the delay durations, where the delays are exact in the no-wait situation.

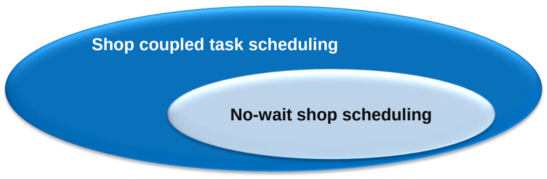

Figure 3: Shop coupled task scheduling and no-wait shop scheduling.

1.3 Paper contribution

The contributions of this paper include (1) presenting a comprehensive review of available studies on scheduling problems with coupled tasks, both in the single-machine and shop environments, (2) proposing several sets of benchmark instances for the problem, (3) investigating available models for the coupled task scheduling problem and evaluating their performance over the generated instances, and (4) proposing a new formulation for the coupled task problem by adapting a state-of-the-art mathematical model for the no-wait flow shop problem and improving several existing models.

1.4 Paper organization

We organize the rest of the paper as follows: In Section 3 we review the studies discussing the single-machine environment and in Section 4 we review the studies in the shop setting. In Section 5 we generate comprehensive sets of benchmark instances for the problem. We discuss mathematical programming models for the problem in Section 6 and evaluate their performance over the generated instances in Section 7. We discuss potential topics for future research in Section 8.

2 Problem statement and notation

We formally formulate the single-machine coupled task scheduling problem as follows: There is a set $N = \{1, 2, \ldots, n\}$ of jobs, indexed by j, where two tasks are associated with each job $j \in N$. The first (initial) task and its processing time are denoted by a_j while the second (completion) task and its processing time are denoted by b_j. As a result, there is a set $H = \{1, 2 \ldots, 2n\}$ of tasks, indexed by h. Both tasks have known durations and the second task of a job must be started with a delay after the completion of the first task. We denote the delay duration between the tasks of job j as L_j, as illustrated in Figure 4. In the context of coupled task scheduling, it is typical to present the parameters for job j via a triple (a_j, L_j, b_j).

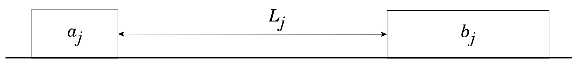

Figure 4: A coupled task job.

In the shop setting, the set N of jobs need to be processed on a set $M = \{1, 2, \ldots, m\}$ of different machines, indexed by k. The coupled task assumption in the shop setting represents the situation where exact delays are considered between each two consecutive tasks of the same job. The processing time of job j's task on machine k is denoted by $p_{k,j}, \forall k \in M, \forall j \in N$, and the delay duration of job j after being processed on machine k is represented by $L_{k,j}$. For simplicity, in the two-machine setting, we replace $p_{1,j}$ and $p_{2,j}$ by a_j and b_j, respectively. Likewise, we replace $L_{k,j}$ by L_j because there is only one delay associated with every job.

Throughout this paper we use the standard three-field notation $\alpha|\beta|\gamma$ for describing scheduling problems proposed by Graham et al. (1979), where α, β, and γ represent the scheduling environment, job characteristics, and performance criterion. Regarding the field α, we use three entries in this paper, namely 1, Fm, and Om, to denote the single-machine, flow shop (with m machines), and open shop (with m machines) environments, respectively.

As regards the β field, (a_j, L_j, b_j) represents the coupled task characteristic in the single-machine setting, which can be modified based on the values of a_j, L_j, and b_j. On the other hand, in the shop setting, the coupled task problem is represented by L_j. Two other characteristics of the jobs include *prec* and G_c, indicating precedence constraints and compatibility constraints, respectively. These constraints can be in different forms, e.g., *star*, *chain* etc. The situations where only permutation schedules are considered is shown by *perm* and where a fixed-job-sequence is stipulated is denoted by fjs.

With respect to the field γ, let d_j, C_j, L_j, E_j, and T_j denote the due date, completion time, lateness, earliness, and tardiness of job j, respectively, where $L_j = C_j - d_j$, $E_j = \max_j\{d_j - C_j, 0\}$, and $T_j = \max_j\{C_j - d_j, 0\}$. The most commonly used objective function for the coupled task scheduling problem is the schedule length, i.e., the makespan, which is denoted as C_{max}, i.e., the maximum job completion. Other objective functions include the total completion time $\sum_j C_j$, maximum lateness L_{max}, earliness-tardiness penalties $\sum_j (E_j + T_j)$, and total energy consumption TEC (as in Liu et al. (2017)).

Tables 2 and 3 summarize the notation used in this paper, as well as the entries used in the three-field notation.

3 The single-machine scheduling problem

The majority of the studies on the coupled task scheduling problem are in the context of exact delays in the single-machine environment. Given a set N of jobs, job $j \in N$ is represented by the triple (a_j, L_j, b_j). Preemption is not allowed, implying once the operation of a task is started, it must be completed with no interruption. However, the tasks of other jobs can be processed during the delay period. Next, we first discuss studies considering the objective function of minimizing the makespan and then we review works that possess the cyclic characteristic.

3.1 Makespan

Almost all the available studies in the single-machine environment consider the objective function of minimizing the makespan, i.e. C_{max}. There are several studies, however, that consider the cyclic characteristic, which we review in Section 3.2. It should be noted that in the single-machine environment minimizing the idle times of the machine is equivalent to minimizing the makespan.

Shapiro (1980) made the first attempt to explore the computational complexity of the single-machine coupled task scheduling problem. He showed that the problem is equivalent to a job shop, where n jobs are to be scheduled on two machines (e.g., M_1 and M_2) with the following characteristics: (1) every job requires three tasks, where the first task is performed on M_1, the second task, i.e., the delay period, on M_2, and the third task on M_1; (2) machine M_1 may only process one task at a time, however, M_2 has infinite processing capacity; and (3) no waiting time between every pair of tasks of a job is permitted, i.e., once the processing of a job is started, all its tasks must be finished with no delay. Hence,

Table 2: Mathematical notation used in this paper.

Notation	Description
Set:	
N	Set of jobs, $N = \{1, \ldots, n\}$, indexed by j.
H	Set of tasks, $H = \{1, \ldots, 2n\}$, indexed by h, where H_{2j-1} and H_{2j} are the initial and completion tasks of job j.
M	Set of machines, $M = \{1, \ldots, m\}$, indexed by k.
Θ	Set of time slots, $\Theta = \{1, \ldots, T\}$, indexed by t.
Parameter:	
n	Number of jobs, and also number of positions in the shop setting.
m	Number of machines.
T	Number of time slots.
a_j	Processing time of the initial task of job j, $a_j \in \mathbb{Z}^+$, also processing time of job j on the first machine in a two-machine shop setting.
b_j	Processing time of the completion task of job j, $b_j \in \mathbb{Z}^+$, also processing time of job j on the second machine in a two-machine shop setting.
L_j	Duration of the delay of job j, $L_j \in \mathbb{Z}^+$, in the single-machine or two-machine shop setting.
d_j	Due date of job j.
$p_{k,j}$	Processing time of job j on machine k, $p_{k,j} \in \mathbb{Z}^+$, in the m-machine shop setting.
$L_{k,j}$	Duration of the delay of job j after processing on machine k, $L_{k,j} \in \mathbb{Z}^+$, in the m-machine shop setting.
Decision variable:	
s_j	Starting time of job j; similarly, s_h is the starting time of task h.
C_j	Completion time of job j; similarly, $C_{j,k}$ (or $C_{i,k}$) is the completion time of job j (or job in position i) on machine k.
E_j	Earliness of job j: $E_j = \max_j\{d_j - C_j, 0\}$; similarly, E_i is the earliness of job in position i.
T_j	Tardiness of job j: $T_j = \max_j\{C_j - d_j, 0\}$; similarly, T_i is the tardiness of job in position i.
$I_{i,k}$	Idle time of machine k before processing job in position i (see Model F2 in Section 6.2).
$x_{j,t}$	Takes the value of 1 if job j starts at time slot t and 0 otherwise, $x_{j,t} \in \{0,1\}, \forall j \in N, t \in \Theta$ (see Model S1 in Section 6.1).
$x_{h,h'}$	Takes the value of 1 if task h' starts after task h in the sequence and 0 otherwise, $x_{h,h'} \in \{0,1\}, \forall h, h' \in H$ (See Model S3 in Section 6.1).
$x_{i,j}$	Takes the value of 1 if job j is assigned to the position i and 0 otherwise, $x_{i,j} \in \{0,1\}, \forall i, j \in N$ (see Models F1, F2, and F3 in Section 6.2).
$x_{j,j'}$	Takes the value of 1 if job j' is placed immediately after job j in the sequence, and 0 otherwise, $x_{j,j'} \in \{0,1\}, \forall j, j' \in N$ (see Model F4 in Section 6.2).

we derive the NP-hardness of the problem from that of the two-machine job shop problem. Shapiro (1980) also proposed the first heuristic algorithms for the problem. He labelled his three simple heuristics as "sequencing", "nesting", and "fitting" as follows:

The sequencing heuristic, also known as "interleaving", constructs an ordered subset of r jobs such that the completion tasks are sequenced for processing in the same order as the initial tasks are scheduled. In fact, the procedure finds as many jobs as their initial tasks can be processed successively with no overlap in their completion tasks. Figure 5a shows the operations of the sequencing heuristic for two jobs j and j'. The nesting heuristic is similar to the sequencing heuristic; however, the completion tasks are processed in the reverse order of the initial tasks. In particular, this procedure allows the jobs to be processed during the delay periods of the previous jobs. Figure 5b shows the nesting heuristic. The fitting heuristic allows the user to specify a priority order for the jobs. This is almost an interactive scheduling procedure, which is only applicable to instances with a small number of jobs.

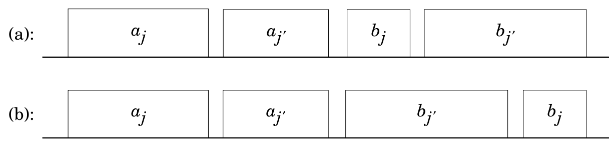

Figure 5: (a): Interleaving jobs j and j', and (b): nesting jobs j and j'.

Orman and Potts (1997) provided a comprehensive study of the complexity issues of the problem. They concluded the NP-hardness for certain cases, which we review in Section 3.1.1. An important result of their study is the symmetry property for the problem. Specifically, they showed that the makespan minimization problem defined by (a_j, L_j, b_j) is equivalent to the one defined by (b_j, L_j, a_j). In other words, the two problems are identical. They called the latter the "reverse" of the former.

Sherali and Smith (2005) studied two variants of the problem. The first variant aims at maximizing the sum of the weights of the jobs that are completed before the time T_{max}. In other words, the model decides which jobs to be scheduled before T_{max} and which ones to be rejected, so as to maximize the total weight. The second variant is the makespan minimization problem. Through a reduction from the 3-Partition problem, they showed that both variants are strongly NP-hard. They proposed two methods to formulate the two variants, namely discretization formulation, in which a time horizon is discretized into unit time slots, and the problem is then to determine the starting time slots for the first tasks of all the jobs, while all the jobs must complete before a time T, and continuous formulation, where the

Table 3: The entries used in this paper for the three-field notation ($\alpha|\beta|\gamma$).

Notation	Description
α:	
1	Single machine.
Fm	Flow shop with m machines.
Om	Open shop with m machines.
β:	
(a_j, L_j, b_j)	Coupled tasks in the single-machine setting.
L_j	Coupled tasks in the shop setting.
$prec$	Precedence constraints, which may be in the special form *in/outtree* or *chain*.
G_c	Compatibility graph, which may be in the special form *star* or *chain*.
$perm$	Permutation.
fjs	Fixed-job-sequence.
γ:	
C_{max}	Makespan.
$\sum_j C_j$	Total completion time.
L_{max}	Maximum lateness.
$\sum_j (E_j + T_j)$	Earliness-tardiness penalties.
TEC	Total energy consumption.

jobs can start at any time.

Condotta and Shakhlevich (2012) studied the complexity of a restricted version of the problem, where the sequence for either the initial or completion tasks of all the jobs is given. They proved that obtaining the optimal solution, given a sequence for the initial tasks, is NP-hard in the strong sense, even if all the jobs have unit execution times (UET). Note that due to the symmetry property of the coupled task scheduling problem, the NP-hardness result holds when the sequence for the completion tasks is given.

Ageev and Kononov (2007) provided an approximation algorithm for the problem. In particular, they proposed an algorithm that works by ordering the jobs/tasks in non-increasing order of $(a_j + L_j)$, i.e., LPT$_{(a_j+L_j)}$, and showed that it is a 3.5-approximation algorithm for the problem. This job ordering has a time complexity of $O(n \log n)$, where n is the number of jobs. Li and Zhao (2007) defined the "singleton" job as the job that can neither be interleaved nor nested. Particularly, job j is a singleton if it cannot be nested within any other job $j' \in N$, nor j' can be nested within j, i.e., in the delay period, nor they can be interleaved with each other. They observed that such jobs can be "appended" (appending refers to scheduling a job without any interleaving or nesting moves) one after another at the end of an optimal schedule of the remaining jobs, i.e., all the jobs excluding the singleton ones. They also presented a lower bound on the makespan as follows:

$$C^*_{max} \geqslant \max \Big\{ P_a + P_b, \max_j (a_j + L_j + b_j), P_a + \min L_j, P_b + \min L_j \Big\}, \tag{1}$$

where $P_a = \sum_j a_j$, $P_b = \sum_j b_j$, and C^*_{max} is the optimal makespan.

Li and Zhao (2007) proposed the first meta-heuristic, i.e., a Tabu Search (TS) algorithm. In order to use permutation representations for generating neighbourhoods, they applied a greedy heuristic, which builds a non-permutation schedule from a given permutation one. The heuristic applies the interleaving, nesting, and appending operations for scheduling a job at the earliest possible time. The neighbourhood procedure consists of removing a fixed number of consecutive jobs and inserting them in different positions in the sequence. The TS algorithm of Condotta and Shakhlevich (2012) includes a neighbourhood that removes a job from its position and inserts it in another position. They used the disjunctive graph model for job re-insertion. In a disjunctive graph, the nodes represent tasks and the arcs show the precedence relations between the tasks, where the arcs can be either conjunctive or disjunctive. The conjunctive arcs represent the precedence relations between the tasks of the same job. The disjunctive arcs, however, show the precedence requirements between the tasks of different jobs. Their TS algorithm outperforms the adapted heuristic of joint decompose local search proposed for the problem with flexible delays (Potts and Whitehead, 2007) and their own greedy dispatching rule.

Békési et al. (2014) proposed a branch-and-bound (B&B) algorithm. They developed two mathematical programming formulations based on linear ordering variables. They also implemented two additional models, a time-indexed model by utilizing the discretized model of Elshafei et al. (2004) and the continuous model of Sherali and Smith (2005). They used the four models to evaluate the performance of their B&B algorithm. They concluded that for instances with UET tasks, the time-indexed model performs better, while for the remaining instances, the B&B algorithm obtains higher quality solutions and in shorter times than optimizing the models by solvers .

3.1.1 Restrictions on the duration

In their study, Orman and Potts (1997) classified the problem with respect to the restrictions on the durations of the tasks and the delay period. They showed that many special cases in this regard are NP-hard. For example, the case $(a_j = L_j = b_j)$ is NP-hard in the strong sense, so the cases $(a_j, L_j = b_j)$, $(a_j = b_j, L_j)$, and $(a_j = L_j, b_j)$ are also NP-hard in the strong sense. They also showed the strong NP-hardness of the case (a_j, L, b), implying the same result for the cases (a, L_j, b), (a, L, b_j), (a_j, L_j, b), (a, L_j, b_j), and (a_j, L, b_j). Even when the initial and completion tasks are equal, i.e., (p, L_j, p) for some constant p, they proved that the case is strongly NP-hard (the same applies to the case (a, L_j, b)). They proposed two optimal algorithms with an $O(n)$ time complexity for the cases (p, p, b_j) and (p, L, p). The former implies that the special cases (a_j, p, p), (p, p, b), and (a, p, p) can also be optimally solved by the same algorithm.

Overall, the study of Orman and Potts (1997) shows that the problem is strongly NP-hard even if two out of the three fields, i.e., (a_j, L_j, b_j), are equal for all the jobs. There is one special case, however, whose complexity status has remained open to date, where all the jobs have the same tasks and delay, i.e., (a, L, b). This special case is called the "identical" problem.

Ahr et al. (2004) also studied the identical problem. The assumed two conditions. First, $a \geqslant b$ because the problem is equivalent if $b \geqslant a$, due to the reverse property discussed earlier. Second, $a < L < (n-1)a$; otherwise, a simple greedy heuristic yields the optimal solution. Under the two assumptions, they proposed a dynamic programming algorithm with an $O(nr^{2L})$ time complexity, where $r \leqslant \sqrt[a-1]{a}$ (note that $\sqrt[a-1]{a}$ approaches 1 when a increases). This algorithm has a linear time complexity in the number of jobs only when L is fixed; however, the complexity status of the problem still remains open. Studying the identical problem, Baptiste (2010) showed that it can be solved by an algorithm in $O(v^{2v+5})$, where $v \leqslant (a^{L/a-1})^{a^{L/a-1}}$. For fixed a, L, and b, his algorithm has an $O(\log n)$ time complexity, so is linear in the number of jobs. This implies that the computational complexity of the identical problem still remains open for arbitrary inputs a, L, and b. Baptiste (2010) also showed that if all the processing times take integer values, then an optimal schedule includes integer starting times.

Arbitrary delays make the problem hard to solve, even with UET tasks. Yu et al. (2004) showed the strong NP-hardness of the two-machine flow shop scheduling problem with exact delays and UET tasks. Based on the NP-hardness of the flow shop problem, they implied that the single-machine coupled task scheduling problem with UET tasks is also strongly NP-hard.

Several studies develop approximation algorithms and ratios for special cases of the problem. For example, Ageev and Kononov (2007) proposed an $O(n \log n)$ algorithm based on $\text{LPT}_{(a_j+L_j)}$, and showed that it is a 3-approximation when $a_j \leqslant b_j$ or $b_j \leqslant a_j$, and a 2.5-approximation if $a_j = b_j$ Importantly, they proved that, for any $\varepsilon > 0$, the existence of a $(2-\varepsilon)$-approximation algorithm for the single-machine case implies that $P = NP$, even if $a_j = b_j$. Ageev and Baburin (2007) studied the case of UET tasks. They proposed an $O(n \log n)$ algorithm based on non-decreasing ordering of L_j, i.e., SPT_{L_j}, and showed that it is a 1.75-approximation. Békési et al. (2009) later argued that there are some flaws in the approximation algorithm of Ageev and Baburin (2007). They re-calculated the ratio as $\frac{28}{19} \leqslant \rho \leqslant \frac{7}{4}$. Ageev and Ivanov (2016) studied the case with equal delay, i.e., (a_j, L, b_j). They showed that the $\text{LPT}_{(a_j+L_j)}$ algorithm leads to ratios of 2 when $a_j \leqslant b_j$, and 1.5 when $a_j = b_j$. Also, they showed that the 2-approximation algorithm provides the same ratio for the identical problem. In addition, for any $\varepsilon > 0$, the existence of a $(1.25 - \varepsilon)$-approximation algorithm implies $P = NP$, even when $a_j = b_j$. They proposed an $O(n \log n)$ constructive algorithm with a 3-approximation ratio.

Li and Zhao (2007) studied the special case where the initial and completion tasks, and the delay have the same value for every job, i.e., (p_j, p_j, p_j). For this case, denoted as the "stretched" case (Darties et al., 2016), they showed that no two jobs can be interleaved, and that any schedule without forced idle time yields a makespan value that is no greater than 1.5 times the optimal. For the problem with equal delays and equal completion tasks, i.e., (a_j, L, b), they concluded that no nesting is possible for any pair of jobs. They showed that a schedule with makespan at most three times of the optimal one can be constructed in linear time.

3.1.2 Restrictions on the job sequence

Restrictions on the job sequence include precedence relationships, compatibility, and fixed job sequences. In this section we discuss research in each area.

Precedence constraints. Precedence constraints model jobs' dependency, i.e., when a job is required to be processed before another job. Precedence constraints are usually defined by using a precedence graph, which can be in a general form, or in a special form such as a chain or a star. The coupled task problem with precedence constraints is usually harder to solve. For example, while the problem with equal initial and completion tasks, and equal delay, i.e., (p, L, p), is polynomially solvable (Orman and Potts, 1997), if precedence constraints are included, the problem is not necessarily solvable in polynomial time.

Blazewicz et al. (2010) investigated the problem with UET tasks and two types of precedence constraints, namely strict and weak precedence constraints. For two jobs j and j', the strict precedence constraint $N_j \prec N_{j'}$ (N_j precedes $N_{j'}$) means $b_j \prec a_{j'}$, i.e., b_j must be completed prior to the start of $a_{j'}$. On the other hand, the weak precedence constraint $N_j \prec N_{j'}$ means $a_j \prec a_{j'}$ and $b_j \prec b_{j'}$, i.e., a_j must be completed prior to the start of $a_{j'}$ and b_j must be completed prior to the start of $b_{j'}$. They showed that the problem with strict precedence constraints is NP-hard in the strong sense, when L is arbitrary. However, for the special case where the delay duration is equal to two and the precedence graph is an in/out-tree (the in-tree (out-tree) precedence constraints occur when every job has at most one immediate successor (predecessor)), the problem can be solved in $O(n)$ time. For this reason, they developed an algorithm that operates by utilizing the rule proposed by McNaughton (1959) for the parallel-machine scheduling problem.

Ecker and Tanaś (2012) studied the problem with UET tasks, equal delays, and strict precedence constraints in the form of chains. Based on McNaughton's rule, they proposed an algorithm that solves the problem in $O(n \log n)$ time if L is a fixed constant. Blazewicz et al. (2012) studied a very special case of the problem where $L = 4$. They proposed a greedy heuristic and an optimal algorithm with an $O(n \log n)$ time complexity. As a generalization of the two algorithms, they presented an approximation algorithm that works for any $L = 2r, r \in \mathbb{N}$, and runs in $O(n \log n)$ time. The solution delivered by the approximate algorithm is at most L times worse than that of the optimal.

Compatibility constraints. Two jobs j and j' are compatible if at least one of the tasks of either of jobs can be processed during the idle time of the other job. Compatibility of jobs can be represented by a compatibility graph, in which an edge is associated with every two compatible jobs. Simonin (2009) showed that the identical coupled task problem with compatibility constraints is NP-hard.

Simonin et al. (2011b) showed that the identical problem can be solved in polynomial time if $L < a + b$. However, it is NP-hard if $L = a + b$. They proposed a ρ-approximation algorithm, where ρ depends on the values of a and b ($a > b$); however, $1.25 \leqslant \rho \leqslant 1.5$. Simonin et al. (2011a) investigated the identical problem with precedence constraints, incompatibility graph, and UET tasks. They considered equal delays for all the jobs and showed that the problem is NP-hard when $L \geqslant 3$. They presented an approximation algorithm with a ratio of $\frac{L+6}{6} - \frac{1}{2(L+2)} + \frac{L+3}{6n(L+2)}$ (it is trivial though if $L = 1$). When $L = 2$, Simonin et al. (2013) studied the problem under the conditions of with and without"triangles connectivity" in the compatibility graph. They showed that the problem is NP-hard under both conditions, and presented a $\frac{10}{9}$ ($\frac{13}{12}$)-approximation algorithm for the existence (lack) of triangles. Darties et al. (2016) studied the case (p_j, p_j, p_j) and denoted p_j as the stretch factor of job j. They showed that the problem can be solved in $O(n^3)$ time when the compatibility graph is a chain and it is NP-hard when the graph is a star. For the problem with a chain or a 2-stage bipartite compatibility graph, they proposed $\frac{7}{6}$ and $\frac{13}{9}$-approximation algorithms (a 2-stage is a bipartite graph that has three sets of disjoint nodes. The first set is disjoint, but connected to the second set, which itself is also disjoint but connected to the third set).

Recently, Bessy and Giroudeau (2018) investigated the parameterized complexity of the coupled task scheduling problem with compatibility graphs. Given a fixed due date d for all the jobs, the parameterized analysis includes whether a schedule in which at least a fixed number of jobs are completely processed before d exists. They proved that the problem is fixed-parameter tractable (FPT) when the total duration of every job, i.e., $a_j + L_j + b_j, \forall j$, is bounded by a constant, and also showed that the problem is W[1]-hard otherwise. Particularly, they showed that for every $\zeta \geqslant 4$, where ζ is an upper bound on the total duration of a job, i.e., $a_j + L_j + b_j \leqslant \zeta$, an FPT algorithm exists for the coupled task scheduling problem with a time complexity of $2^{O(k)} n^{2\zeta} \log^2 n$, where $k \leqslant n$.

Fixed job sequence. Hwang and Lin (2011) studied the problem subject to a fixed-job-sequence (*fjs*), assuming weak precedence constraints. Note that the problem with the *fjs* assumption and strict precedence constraints is trivial. They argued that although *fjs* implies a pre-assigned job sequence, the problem remains a sequencing one due to the interleaving jobs. They developed a procedure to construct a schedule for a given sequence, and showed that if such a sequence is feasible, then the schedule has the minimum makespan among all the feasible sequences. The procedure also concludes the infeasibility of a given sequence in $O(n^2)$ time. While the complexity status of the problem is open, they proposed polynomial-time procedures for three special cases of the problem, namely (p_j, p_j, p_j), (p, p, b_j), and (p, L, p).

3.2 Cyclic scheduling

Ahr et al. (2004) investigated an interesting structure of the optimal schedule for the identical problem when the number of jobs theoretically approaches infinity. They argued that an optimal schedule consists of three parts, namely an "initial" segment, followed by a certain number of repetition of cycles (the "middle" part), and a "finishing" segment. They stated that the middle segment repetitions must contain the minimum mean cycle, i.e., the cycle with the smallest mean weight value. For the special case where $b = 1$, they conjectured a formula for the minimum mean cycle time. They also presented optimal ratios and mean cycle times for certain small instances.

This motivates Brauner et al. (2009) to link the problem to a class of cyclic production scheduling problems, namely the one-machine no-wait robotic cell problem. They showed that minimizing the makespan for the single-machine identical coupled task scheduling problem is equivalent to maximizing the throughput rate in a one-machine no-wait robotic cell with the characteristics of having an input station (IN), an output station (OUT), and one machine (M). Raw material and finished products can be stored in IN and OUT with infinite capacity. Any number of products can be treated simultaneously by machine M. A single transporter that has unit capacity carries materials between IN, M, and OUT. To obtain a production pattern, they adapted the algorithm of Ahr et al. (2004).

Lehoux-Lebacque et al. (2015) continued work on the identical problem in the cyclic case. They considered the objective function of maximizing the throughput, i.e., $\frac{T(C)}{N(C)}$, where $T(C)$ is the cycle time and $N(C)$ is the number of tasks in the cycle C. It should be noted that maximizing the throughput is equivalent to minimizing the mean cycle time $(\frac{N(C)}{T(C)})$. They studied the problem for $a > b$, as for $b > a$ the reverse property holds and for $a = b$ it is trivial. Likewise, the condition $L > a + b$ is imposed; otherwise, the optimal cycle can be easily obtained. They studied all the possible patterns that may occur in a cycle, and presented an algorithm with a time complexity of $O((\log L)^2)$ to find the optimal cycle. In addition, considering $\beta = \lfloor \frac{L}{a+b} \rfloor$, if $L - \beta(a + b) \leqslant a$, the optimal mean cycle is the triple $(a, \frac{L-\beta(a+b)}{\beta+1}, b)$. Such results let them arrive at the conjectures of Ahr et al. (2004). The results of Lehoux-Lebacque et al. (2015) enable the finding of the optimal schedule for the middle segment when n is very large. However, obtaining the optimal schedule for the initial and finishing segments is not trivial.
Table 4 summarizes the reviewed studies and their results for the single-machine coupled task scheduling problem.

Table 4: Summary of of the reviewed literature on the single-machine coupled task scheduling problem.

Problem	**Complexity and solution approach**	**Comment**	**Reference**
$1\|\beta\|C_{max}$ where β is:			
(a_j, L_j, b_j)	Strongly NP-hard, nesting and interleaving heuristics	Experiments with simulated data	Shapiro (1980)
	Strongly NP-hard, mathematical formulations	Also holds for weighted model with $\sum_j w_j$, experiments with normal data	Sherali and Smith (2005)
	3.5-approximation algorithm with $\text{LPT}_{(a_j+L_j)}$	$O(n \log n)$	Ageev and Kononov (2007)
	Tabu Search algorithm	Experiments with uniform data	Li and Zhao (2007)
	Strongly NP-hard even if the sequence for initial tasks is given, Tabu Search algorithm	Experiments with uniform data	Condotta and Shakhlevich (2012)
	B&B algorithm	Experiments with uniform and normal data	Békési et al. (2014)
$(a_j = L_j = b_j)$	Strongly NP-hard	Also holds for $(a_j, L_j = b_j)$, $(a_j = b_j, L_j)$, $(a_j = L_j, b_j)$	Orman and Potts (1997)
(a_j, L, b)	Strongly NP-hard	Also holds for (a, L_j, b), (a, L, b_j), (a_j, L_j, b), (a, L_j, b_j), (a_j, L, b_j)	Orman and Potts (1997)
(p, L_j, p)	Strongly NP-hard	Also holds for (a, L_j, b)	Orman and Potts (1997)
(p, p, b_j)	Optimal with a constructive algorithm	$O(n)$, also holds for (a_j, p, p)	Orman and Potts (1997)
(p, L, p)	Optimal with a constructive algorithm	$O(n)$, also holds for (a, p, p), (p, p, b) and (p, p, p)	Orman and Potts (1997)
(a, L, b)	Open problem		Orman and Potts (1997)
	$O(nr^{2L})$ algorithm where $r \leqslant \sqrt[a-1]{a}$	Linear in n if L is fixed, experiments with random data	Ahr et al. (2004)
	$O(\log n)$ algorithm	Linear in n if a, b, and L are fixed	Baptiste (2010)
	2-approximation algorithm with $\text{LPT}_{(a_j+L_j)}$	$O(n \log n)$	Ageev and Ivanov (2016)
$(a_j \geqslant b_j, L_j)$	3-approximation algorithm with $\text{LPT}_{(a_j+L_j)}$	$O(n \log n)$, also holds for $(a_j \leqslant b_j, L_j)$	Ageev and Kononov (2007)
$(a_j = b_j, L_j)$	2.5-approximation algorithm with $\text{LPT}_{(a_j+L_j)}$	$O(n \log n)$	Ageev and Kononov (2007)
	(2-ε)-approximation algorithm is NP-hard		Ageev and Kononov (2007)
$(1, L_j, 1)$	ρ-approximation algorithm, where $\frac{28}{19} \leqslant \rho \leqslant \frac{7}{4}$		Békési et al. (2009)
		Experiments with uniform data	Békési et al. (2014)
(a_j, L, b_j)	3-approximation algorithm with a constructive heuristic	$O(n \log n)$	Ageev and Ivanov (2016)
$(a_j \leqslant b_j, L)$	2-approximation algorithm with a constructive heuristic	$O(n \log n)$	Ageev and Ivanov (2016)
$(a_j = b_j, L)$	1.5-approximation algorithm with a constructive heuristic		Ageev and Ivanov (2016)
	1.25-approximation algorithm is NP-hard	Also holds for $(a_j \leqslant b_j, L)$ and (a_j, L, b_j)	Ageev and Ivanov (2016)
(p_j, p_j, p_j)	1.5-approximation algorithm with any feasible schedule		Li and Zhao (2007)
(a_j, L, b)	3-approximation algorithm with a constructive heuristic		Li and Zhao (2007)
$1\|\beta, prec\|C_{max}$ where β is:			
$(1, L, 1)$	Strongly NP-hard	L is a part of input	Blazewicz et al. (2010)
$(1, 2, 1)$, $in/out-tree$	Optimal with a constructive algorithm	$O(n)$	Blazewicz et al. (2010)
$(1, L, 1)$, $chain$	Optimal with an algorithm based on McNaughton's rule	$O(n \log n)$	Ecker and Tanaś (2012)
$(1, 4, 1)$, $chain$	Greedy heuristic	$O(n \log n)$	Blazewicz et al. (2012)
	Optimal with an algorithm based on McNaughton's rule	$O(n \log n)$	
$(1, 2r, 1)$, $chain$, $r \in \mathbb{N}$	An approximation algorithm based on McNaughton's rule	$O(n \log n)$, Experiments with random data	Blazewicz et al. (2012)
$1\|\beta, G_c\|C_{max}$ where β is:			
(a, L, b)	NP-hard	Also holds for $(a_j = L_j = b_j)$	Simonin (2009)
(a, L, b), $L < a + b$	Optimal with a maximum matching	$O(r\sqrt{n})$, where r is the number of edges in G_c	Simonin et al. (2011b)
(a, L, b), $L = a + b$	NP-hard		Simonin et al. (2011b)

	$\frac{3a+2b}{2a+2b}$-approximation algorithm, $1.25 \leqslant \rho \leqslant 1.5$	Upper bound decreases to $\frac{1+\sqrt{3}}{2}$ for some special cases	
$(1, L, 1), L \geqslant 3, prec$	NP-hard	Optimal with a maximum matching when $L = 1$	Simonin et al. (2011a)
	$\frac{L+6}{6} - \frac{1}{2(L+2)} + \frac{L+3}{6n(L+2)}$-approximation algorithm		
$(1, 2, 1)$	$\frac{10}{9}$-approximation algorithm for G_c in presence of triangles	$\frac{13}{12}$-approximation algorithm for G_c in lack of triangles	Simonin et al. (2013)
(p_j, p_j, p_j), *chain*	$O(n^3)$		Darties et al. (2016)
(p_j, p_j, p_j), *star*	NP-hard	Also holds for 2-stage bipartite G_c	Darties et al. (2016)
	$\frac{7}{6}$-approximation algorithm	Also $\frac{13}{9}$-approximation algorithm for 2-stage bipartite G_c	Darties et al. (2016)
$k - parameterized, d$	FPT with $2^{O(k)} n^{2\zeta} \log^2 n$ time when ζ-bounded with $\zeta \geqslant 4$	W[1]-hard otherwise	Bessy and Giroudeau (2018)
$1\|\beta, fjs\|C_{max}$ where β is:			
(a_j, L_j, b_j)	Optimal or showing infeasibility	$O(n^2)$	Hwang and Lin (2011)
(p_j, p_j, p_j)	Optimal with a constructive algorithm	$O(n)$	Hwang and Lin (2011)
(p, p, b_j)	Optimal with a forward dynamic programming	$O(n)$	Hwang and Lin (2011)
(p, L, p)	Optimal with a constructive algorithm	$O(n)$, symmetric to $1\|(p, L, p)\|C_{max}$	Hwang and Lin (2011)

4 The shop scheduling problem

Almost all the published studies on the coupled task scheduling problem in the shop environment are conducted in the flow shop context. Therefore, we discuss the flow shop environment in this section. In Section 4.6, we present research in other shop scheduling environments.

The classical flow shop scheduling problem includes a set N of jobs to be processed on a set M of different machines. Job j consists of a set of m tasks (operations) to be processed in the "same" sequence on the m machines.

The coupled task scheduling in the flow shop environment deals with the situation in which there are "exact" delays between every pair of consecutive tasks of a job. All the tasks have known processing times and all the delays have known durations. In addition, the delay durations are strict, i.e., once the delay period is elapsed, the processing of the next task must immediately start. No two tasks can be processed at the same time on one machine and preemption is not allowed. However, tasks of other jobs can be processed during the delay period. Figure 6 shows a schematic of the m-machine flow shop coupled task scheduling problem.

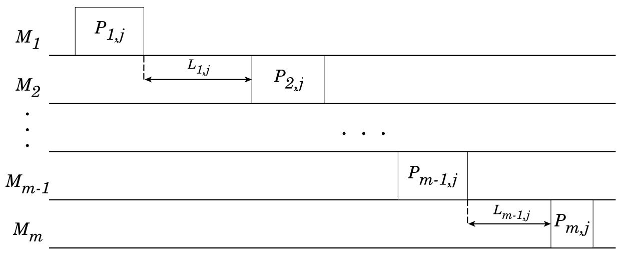

Figure 6: Schematic of the m-machine flow shop coupled task scheduling problem.

The flow shop coupled task problem is a generalization of the no-wait flow shop scheduling problem. In other words, the no-wait flow shop problem is a special case of the flow shop coupled task problem, where $L_{k,j} = 0, k = 1, \dots, m-1, \forall j \in N$. Recall that in the no-wait flow shop, once the processing of a job is started, its tasks must be processed on all the machines (from the first to the last) without any interruption. Next, we review research in the context of the flow shop coupled scheduling problem according to the performance criterion. The major performance criteria adopted in the literature include the makespan, total completion time, maximum lateness, and earliness and tardiness.

4.1 Makespan

The problem of minimizing the makespan in a two-machine flow shop, i.e., $F2||C_{max}$, is one of the first and well studied scheduling problems. Johnson (1954) proposed an $O(n \log n)$ algorithm to find the optimal schedule for the problem, which is a permutation one. A permutation schedule is any schedule in which the processing orders of the jobs on all the m machines are the same. For the same two-machine problem with exact delays, i.e., $F2|L_j|C_{max}$, there exists an optimal schedule, which is also a permutation one, if all the delays are equal, i.e., $L_j = L$, and the optimal schedule is obtained in $O(n \log n)$ time (Leung et al., 2007). However, they did not discuss how the algorithm works. We observe that the algorithm proposed by Gilmore and Gomory (1964) for the two-machine no-wait flow shop problem is optimal for $F2|L_j = L|C_{max}$ as well. This is because in any sequence for $F2|L_j = L|C_{max}$, the delay duration of the first job leads to shifting the completion task of all the jobs forward, i.e., $C_j^{L_j=L} = C_j^{\text{no-wait}} + L, \forall j \in N$. Moreover, there is no job that can be swapped in order to use this delay duration because all the jobs have an identical delay. Hence, the optimal solution for the no-wait case yields the optimal solution for the problem $F2|L_j = L|C_{max}$, where $C_{max}^{L_j=L} = C_{max}^{\text{no-wait}} + L$.

However, the problem is not trivial if arbitrary delays are considered. For example, Yu et al. (2004) showed that even if two distinct values are considered for the delays, then the two-machine flow shop coupled task problem to minimize the makespan is strongly NP-hard. This result holds even for the case with UET tasks. It should be noted that the optimal schedule for the problem with arbitrary delays is not necessarily a permutation one. Leung et al. (2007) discussed an example for this, which is shown in Figure 7. For two jobs $J_1 = (1, 8, 1)$ and $J_2 = (2, 1, 3)$, the figure shows that the optimal makespan is 10. However, the makespan of the best permutation schedule is 12.

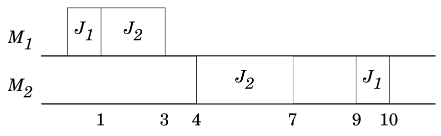

Figure 7: An example showing that the optimal schedule for $F2|L_j|C_{max}$ is not necessarily a permutation one.

Leung et al. (2007) also presented a simple lower bound for $F2|L_j|C_{max}$ as follows:

$$C^*_{max} \geqslant \max\Big\{P_a, P_b, \max_j\{a_j + L_j + b_j\}\Big\}, \tag{2}$$

where $P_a = \sum_j a_j$ and $P_b = \sum_j b_j$.

For the m-machine case $Fm|L_j|C_{max}$, Hamdi and Loukil (2017) proposed a lower bound as follows:

$$C^*_{max} \geqslant \max_{1\leqslant k\leqslant m-1}\Big\{\sum_{j=1}^{n} p_{k,j} + \sum_{k'=1}^{m-1}\min_j\,(p_{k',j} + L_{k',j})\Big\}. \tag{3}$$

For the two-machine flow shop coupled task problem, Ageev and Kononov (2007) presented a two-phase algorithm. The first phase sequences the jobs in non-decreasing order of $a_j + L_j$. The second phase constructs the schedule for that sequence. They showed that this simple algorithm provides a 3-approximation ratio for the problem, and that the ratio is relatively small because, for any $\varepsilon > 0$, the existence of a $(1.5 - \varepsilon)$-approximation algorithm for the problem implies that $P = NP$, even if $a_j = b_j$. They discussed the same algorithm that provides a 2-approximation ratio for the special case where $a_j \leqslant b_j$ or $b_j \leqslant a_j$, $\forall j \in N$. The algorithm can be implemented in $O(n \log n)$ time.

Ageev and Baburin (2007) proposed a two-phase algorithm for the two-machine flow shop coupled task problem with UET tasks. Here, the first phase includes sequencing the jobs in non-decreasing order of L_j. They showed that the algorithm provides a 1.5-approximation ratio for the problem, and has a time complexity of $O(n^2 \log n)$. They did not analyze the tightness of the ratio, which remains as an open question. Both of the algorithms of Ageev and Kononov (2007) and Ageev and Baburin (2007)) are essentially the same when UET tasks are considered because $a_j = 1, \forall j \in N$, and sequencing the jobs in non-decreasing order of $a_j + L_j$ leads to the same order obtained by non-decreasing ordering of L_j. To conclude, for the two-machine flow shop coupled task problem, sorting the jobs in non-decreasing order of $a_j + L_j$ leads to a 3-approximation algorithm for the general case, to a 2-approximation when $a_j \leqslant b_j$ or $b_j \leqslant a_j$, and to a 1.5-approximation when $a_j = b_j = 1, \forall j \in N$.

A common assumption in scheduling research is that the processing times take positive integers. Leung et al. (2007) considered a variant in which at least one positive task is given for every job. In other words, there may be jobs with only one positive task. For convenience, we use F^g to denote this variant and F to denote the classical case. It should be noted that F^g is a generalization of F. In their study, if one of the tasks of a job has zero processing time, the delay of the job also has a zero value. On the other hand, if the delay duration of a job takes a non-zero value, both tasks of the job have positive processing times. However, there might be a job with positive processing times for its two tasks and a zero-duration delay.

Leung et al. (2007) showed that the problem $F2^g|L_j \in \{T_1, T_2\}|C_{max}$, i.e., where the delay can only take two values, is strongly NP-hard because the F^g variant of the two-machine no-wait flow shop problem is strongly NP-hard (Sahni and Cho, 1979). They proposed a 3-approximation algorithm, which is applicable to the F^2 and $F2^g$ problems in $O(n \log n)$ time. The algorithm sequences the jobs in non-decreasing order of L_j and then obtains the schedule for the sequence. They showed that when $a_j \geqslant b_j$, sequencing the jobs in non-decreasing order of $a_j + L_j$ leads to a 2-approximation algorithm. The same argument applies to the $F2^g$ variant. Likewise, sequencing the jobs in non-increasing order of $L_j + b_j$ provides a 2-approximation algorithm for the case $a_j \leqslant b_j$, for both the F and F^g variants. They showed that the ratios are tight.

4.2 Total completion time

Leung et al. (2007) studied the flow shop coupled task problem to minimize the total completion time. They applied the NP-hardness of the two-machine no-wait flow shop scheduling problem to minimize the total completion time (Röck, 1984) to imply that the following two-machine problems are strongly NP-hard: (1) the F variant with equal delays and (2) the F^g variant with arbitrary delays. They also proposed a lower bound on the total completion time C^* on two machines. Let $a_{j_1} \leqslant a_{j_2} \leqslant \cdots \leqslant a_{j_n}$ denote the processing times of the tasks on the first machine, sorted in non-decreasing order of the values. Similarly, let $b_{j_1} \leqslant b_{j_2} \leqslant \cdots \leqslant b_{j_n}$ denote the processing times of the tasks on the second machine, again sorted in non-decreasing order of their values. Let $P'_a = \sum_i (n-i+1)a_{j_i}$, $P'_b = \sum_i (n-i+1)b_{j_i}$, and $L' = \sum_j L_j$, and the lower bound is as follows:

$$C^* \geqslant \max\Big\{P'_a + L' + P_b, \min_j\,(a_j + L_j) + P'_b\Big\}, \tag{4}$$

where C^* is the optimal value of the total completion time.

Another interesting result of their study includes generating an optimal schedule for a special case of the F variant where $a_j = a, b_j = b$, and $a \geqslant b$ by sequencing the jobs in non-decreasing order of L_j. This algorithm, however, is not optimal if $a < b$ and provides a 2-approximation ratio instead.

Huo et al. (2009) provided the major results for the total completion time criterion. They studied the permutation version of the flow shop coupled task problem. They showed that, as for the makespan minimization criterion, the optimal schedule for the two-machine flow shop to minimize the total completion time is not a permutation schedule. This can be shown by the following example. Consider three jobs $J_1 = (1, 5, 3)$, $J_2 = (1, 9, 2)$, and $J_3 = (3, 3, 3)$, where a non-permutation schedule generates the minimum total completion time of 35, whereas that of a permutation one is 38. They argued that forced idle time will never improve a permutation schedule under arbitrary delays. Therefore, they assumed that no forced idle time in any feasible schedule for the problem. Because any feasible schedule for the two-machine no-wait flow shop scheduling problem, which itself is NP-hard, is a permutation schedule, they implied that the permutation flow shop coupled task problem is strongly NP-hard.

They investigated several special cases as well. For the first case, let the jobs be ordered in such a way that $a_j \leqslant a_{j+1}$. If $a_{j+1} + L_{j+1} \geqslant L_j + b_j$ for all $1 \leqslant j \leqslant n-1$, the problem is optimally solved by sequencing the jobs in non-decreasing order of a_j. For the second case, assume that the jobs are ordered in such a way that $b_j \leqslant b_{j+1}$. If $a_1+L_1 = \min_j (a_j + b_j)$ and $a_{j+1} + L_{j+1} < L_j + b_j$ for all $1 \leqslant j \leqslant n-1$, the problem is also optimally solved by sequencing the jobs in non-decreasing order of b_j. The third case is when $\max_j a_j \leqslant \min_{j'} b'_j, L_j = L$, which is also polynomially solvable by a SPT$_{b_j}$-based algorithm that obtains the minimum total completion time from a total of n schedules. Finally, for $F2|perm, L_j, a_j = a, b_j = b|\sum C_j$, sequencing the jobs in non-decreasing order of $a_j + L_j + b_j$ is optimal. The same rule also leads to an optimal solution for $F2|perm, L_j = L, a_i < a_j \rightarrow b_i < b_j|\sum C_j$.

Huo et al. (2009) proposed two meta-heuristics, namely Simulated Annealing (SA) and Tabu Search (TS), for the general case of the problem. According to the computational results, SA performs slightly better than TS in terms of both solution quality and computing time. In addition, they showed that a greedy heuristic, which inserts the jobs one by one such that the completion time of the inserted job has the smallest value, performs better than sorting the jobs in non-decreasing order of any of the following: a_j, b_j, $a_j + L_j$, $L_j + b_j$, and $a_j + L_j + b_j$.

It is noted the problem to minimize the total completion time is computationally less demanding than that to minimize the makespan. For example, $F2|L_j, a_j = a, b_j = b, a \geqslant b|\sum C_j$ and so $F2|L_j, a_j = b_j|\sum C_j$ are polynomially solvable; but $F2|L_j, a_j = b_j = 1|C_{max}$ is NP-hard.

4.3 Maximum lateness

Fondrevelle et al. (2009) studied the m-machine flow shop coupled task problem to minimize the maximum lateness. They considered the problem in the permutation setting, finding that permutation schedules are not dominating even if only one job has a delay greater than 0.

They defined the lateness of a job based on its completion time on any machine $k < m$ and distinguished three different forms of the jobs. They presented a special case of the problem for which the earliest due date (EDD) dispatching rule provides the optimal schedule. In addition, they developed several dominance rules for the two-machine problem and proposed a B&B algorithm. They used the well-known algorithm of Nawaz, Enscore and Ham (NEH) (Nawaz et al., 1983) to build upper bounds, and the EDD rule and an extension of the algorithm of Gilmore and Gomory (1964) (for the no-wait flow shop problem) to generate lower bounds.

4.4 Earliness and tardiness penalties

Hamdi and Loukil (2017) considered permutation schedules for the flow shop coupled task problem to minimize the total earliness and tardiness penalties. They proposed three mathematical programming formulations, namely completion time-based, idle time-based, and starting time-based. They also discussed three lower bounds. The first is a linear relaxation bound and the second one is calculated by summing two bounds on the values of total earliness and tardiness. The third bound is obtained by relaxing the processing times and delays of the jobs. Also, they used simple sequencing rules to produce upper bounds.

4.5 Other variants of the flow shop coupled task scheduling problem

As reviewed, the flow shop coupled task scheduling problem with an exact delay period between every two consecutive tasks is an important variant. In this section we discuss two additional variants with exact delays in the flow shop environment.

Two-machine flow shop problem with coupled tasks on the first machine. Meziani et al. (2018a) studied the two-machine flow shop problem to minimize the makespan, in which the coupled tasks with an exact delay are only considered on the first machine. Specifically, every job consists of three tasks, two coupled tasks on the first machine, followed by a single task on the second machine. The processing times of the tasks of job $j$are represented by (a_j, L_j, b_j, c_j), in which a_j and b_j are the coupled tasks with an exact delay L_j between them, and c_j represents the third task. The task on the second machine can start if the tasks on the first machine are finished. Figure 8 shows a schematic presentation of this problem. It should be noted that the problem is NP-hard because the one on the first machine is NP-hard. Meziani et al. (2018a) proposed a lower bound for the problem as follows:

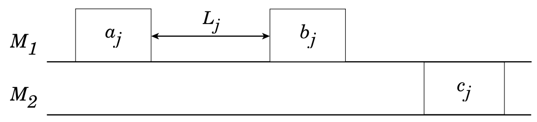

Figure 8: Schematic of the two-machine flow shop problem with coupled tasks on the first machine, and a single task on the second machine.

$$LB = \max\Big\{ \max_j (a_j + b_j) + \min_j c_j, \min_j (a_j + L_j + b_j) + \sum_j c_j, \\ \sum_j a_j + \min_j L_j + \min_j c_j, \sum_j b_j + \min_j L_j + \min_j c_j \Big\}. \tag{5}$$

They also proposed four SPT- and LPT-based heuristics, and two meta-heuristics, namely Particle Swarm Optimization (PSO) and Simulated Annealing (SA). They further designed a hybrid algorithm, in which SA guides PSO to avoid being trapped in low-quality local optima. The performance of the proposed algorithms evaluated on a series of problem instances confirms the superiority of the hybridized algorithm.

Meziani et al. (2018b) extended their previous study and examined the complexity status of several cases. For example, they proved the NP-hardness of the cases (a_j, p, p, c_j) and (p, p, b_j, c_j). For the cases (a, p, p, c_j), (p, p, b, c_j), and (p, L, p, c_j), they proposed optimal solutions by extending the algorithm of Johnson (1954), which has an $O(n \log n)$ time complexity. They also proposed an $O(n^2 \log n)$ time algorithm, by modifying the algorithm of Mitten (1959), to solve the cases (a_j, p, p, c_j) and (p, p, b_j, c_j).

The chain re-entrant flow shop problem. Amrouche and Boudhar (2016) studied a variant in which the coupled tasks are considered in the re-entrant flow shop scheduling environment. In the re-entrant flow shop problem, the jobs may visit any machine more than once. There are various types of the re-entrant shop, such as the chain re-entrant and V-shop. In the chain re-entrant problem, the jobs visit M_1, followed by the rest of the $m-1$ machines, and return back to M_1 for their terminating task. In the V-shop problem, the jobs follow the route $M_1, M_2, \dots, M_{m-1}, M_m, M_{m-1}, \dots, M_2, M_1$. Note that for the two-machine flow shop, the chain re-entrant and V-shop are identical. Amrouche and Boudhar (2016) studied the two-machine chain re-entrant problem to minimize the makesapn, i.e., the jobs pass from the first machine to the second one, and return to the first machine; in other words, in the order M_1, M_2, M_1. Here, the processing times of the tasks of job j can be represented by the triple (a_j, b_j, c_j), where a_j and c_j are processed on the first machine, and b_j is processed on the second machine. We call this problem the "Ch-R". Figure 9 illustrates this.

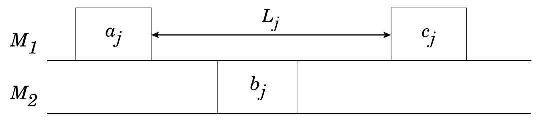

Figure 9: The two-machine chain re-entrant flow shop coupled task problem.

First, they considered an equal delay, i.e., $L_j = L, \forall j$, between two tasks of every job on the first machine, i.e., between the completion of a_j and start of c_j. They showed that even $a_j = c_j = p$ leads to a strongly NP-hard problem. Therefore, the general form of the problem is also strongly NP-hard. They showed that the special case where $a_j > \frac{L}{2}, c_j > \frac{L}{2}$ reduces to the maximum weight matching problem, so is solvable in $O(n^{2.5})$ time. They also showed that the case where $b_j = L_j = L \geqslant a_j + c_j$ is polynomially solvable.

Amrouche et al. (2017) showed that when the delay is equal to the processing time of the task on the second machine, i.e., $b_j = L_j$, the problem is NP-hard in the strong sense. In addition, they showed that the special case of the problem where $b_j = L_j = L$, $a_j + c_j > L$ reduces to the maximum weight matching problem, so is solvable in $O(n^{2.5})$ time. The case where $a_j = a \geqslant c_j$ and $b_j + b'_j \leqslant 2a = L_j$ is also polynomially solvable. For the special case with identical delays, i.e. $L_j = L, \forall j)$, they proposed a heuristic procedure with nine job arrangement rules. Eight rules include sorting the jobs in non-increasing and non-decreasing order of a_j, b_j, c_j, and $a_j + c_j$. The ninth one is a heuristic that generates the schedule with some best fitted batches of the jobs, where each batch consists of a subset of interleaving jobs and a union of all the batches form a complete solution. Their experiments concluded that both the heuristic and sequencing rule in non-increasing order of $a_j + c_j$ perform the best.

Liu et al. (2017) extended the setting of Amrouche and Boudhar (2016) and modelled the time-in-use electricity prices for energy consumption as a bi-criterion re-entrant flow shop. They investigated the two objective functions of minimizing the makespan and minimizing the total energy consumption. They developed a mathematical formulation and proposed two solution methods, namely ε-constraint and Non-dominated Sorting Genetic Algorithm II (NSGA II). They only presented the optimal Pareto front for one small example and conducted no computational experiments to assess the performance of the solution methods.

4.6 The open shop problem

The classical open shop scheduling problem includes a set N of jobs to be processed on a set M of different machines. Job j consists of m tasks, which can be processed in any order on the m machines. The coupled task open shop scheduling problem is similar to its flow shop counterpart with the only difference that the order in which the tasks of a job are processed on the machines is not necessarily the same for all the jobs.

Ageev (2018) argued that because the open shop problem is a generalization of the no-wait environment, and in view of the NP-hardness of the two-machine no-wait open shop problem (Giaro, 2001), the two-machine coupled task open shop problem with exact delays is therefore NP-hard. By using the Partition problem, they proved that the existence of a $(1.5 - \varepsilon)$-approximation algorithm for the special case where $a_j = b_j$ implies $P = NP$, for any $\varepsilon > 0$. They further showed that when the delays can take only two values, there is no approximation algorithm with a ratio better than $(1.5 - \varepsilon)$ for any $\varepsilon > 0$.

Table 5 summarizes the major results for the shop coupled task scheduling problem.

Table 5: Summary of the reviewed literature on the shop coupled task scheduling problem.

Problem	**Complexity and solution approaches**	**Comment**	**Reference**
$F2\|\beta\|C_{max}$, where β is:			
(a_j, L_j, b_j)	Strongly NP-hard		Yu et al. (2004)
	3-approximation algorithm with $\text{SPT}_{(a_j+L_j)}$	$O(n \log n)$	Ageev and Kononov (2007)
	3-approximation algorithm with $\text{SPT}_{(L_j)}$	$O(n \log n)$, Also applicable to $F2^g$	Leung et al. (2007)
$(a_j, L_j \in \{T_1, T_2\}, b_j)$	Strongly NP-hard	Holds for $F2^g$	Leung et al. (2007)
$(a_j \leqslant b_j, L_j)$	2-approximation algorithm with $\text{LPT}_{(L_j+b_j)}$, (likewise, $\text{SPT}_{(a_j+L_j)}$ for $(L_j, a_j \geqslant b_j)$	$O(n \log n)$, Applicable to $F2^g$	Leung et al. (2007)
	2-approximation algorithm with $\text{SPT}_{(a_j+L_j)}$	$O(n \log n)$	Ageev and Kononov (2007)
$(a_j = b_j, L_j)$	(1.5-ε)-approximation algorithm is NP-hard		Ageev and Kononov (2007)
$(1, L_j, 1)$	Strongly NP-hard		Yu et al. (2004)
	1.5-approximation algorithm with $\text{SPT}_{(L_j)}$	$O(n \log n)$	Ageev and Baburin (2007)
(a_j, L, b_j)	Optimal with algorithm of Gilmore and Gomory (1964)	$O(n \log n)$	Leung et al. (2007)
$(Ch-R, b_j \leqslant L)$	Strongly NP-hard	Holds for the case $a_j = c_j = p$	Amrouche and Boudhar (2016)
$(Ch-R, a_j \geqslant \frac{L}{2}, c_j \geqslant \frac{L}{2}, L_j = L)$	Optimal with a maximum weight matching algorithm	$O(n^{2.5})$	Amrouche and Boudhar (2016)
$(Ch-R, b_j = L_j = L, a_j + c_j \leqslant L)$	Optimal with a constructive algorithm	Polynomial	Amrouche and Boudhar (2016)
$(Ch-R, b_j = L_j)$	Strongly NP-hard	Experiments with uniform data	Amrouche et al. (2017)
$(Ch-R, b_j = L_j = L, a_j + c_j > L)$	Optimal with a maximum weight matching algorithm	$O(n^{2.5})$	Amrouche et al. (2017)
$(Ch-R, a_j = a, b_j + b'_j \leqslant 2a, c_j \leqslant a, L_j = 2a)$	Optimal with a constructive algorithm	$O(n \log n)$	Amrouche et al. (2017)
(a_j, L_j, b_j, c_j)	PSO, SA, SA+PSO	Experiments with uniform data	Meziani et al. (2018a)
(a_j, p, p, c_j)	NP-hard	Holds for (p, p, b_j, c_j)	Meziani et al. (2018b)
(a, p, p, c_j)	Optimal with a constructive algorithm	$O(n \log n)$, holds for p, p, b, c_j; p, L, p, c_j	Meziani et al. (2018b)
(a_j, p, p, c)	Optimal with a constructive algorithm	$O(n^2 \log n)$, holds for p, p, b_j, c	Meziani et al. (2018b)
$F2\|\beta\|\sum_j C_j$, where β is:			
(a_j, L_j, b_j)	Strongly NP-hard	Holds for $F2^g$	Leung et al. (2007)
(a_j, L, b_j)	Strongly NP-hard		Leung et al. (2007)
$(a, L_j, b, a \geqslant b)$	Optimal with $\text{SPT}_{(L_j)}$	$O(n \log n)$	Leung et al. (2007)
$(a, L_j, b, a < b)$	2-approximation algorithm with $\text{SPT}_{(L_j)}$	$O(n \log n)$	Leung et al. (2007)
$(a_j, L_j, b_j), perm$	Strongly NP-hard	Experiments with uniform data	Huo et al. (2009)
$(a_j \leqslant a_{j+1} \rightarrow (a_{j+1} + L_{j+1}) \geqslant (L_j + b_j)), perm$	Optimal with $\text{SPT}_{(a_j)}$	$O(n \log n)$	Huo et al. (2009)
$(b_j \leqslant b_{j+1} \rightarrow (a_1 + L_1) = \min_j (a_j + L_j), (a_{j+1}+L_{j+1}) < (L_j+b_j)), perm$	Optimal with $\text{SPT}_{(b_j)}$	$O(n \log n)$	Huo et al. (2009)
$(\max_j a_j \leqslant \min_{j'} b_{j'}, L), perm$	Optimal with a variant of $\text{SPT}_{(b_j)}$	$O(n^2 \log n)$	Huo et al. (2009)
$(a, L_j, b), perm$	Optimal with $\text{SPT}_{(a_j+L_j+b_j)}$	$O(n \log n)$	Huo et al. (2009)
$(L, a_i < a_j \rightarrow b_i < b_j), perm$	Optimal with $\text{SPT}_{(a_j+L_j+b_j)}$	$O(n \log n)$	Huo et al. (2009)
Problems with other objective functions:			
$Fm\|perm, L_j\|L_{max}$	B&B algorithm	Experiments with uniform data	Fondrevelle et al. (2009)
$Fm\|perm, L_j\|\sum_j (E_j + T_j)$	Mathematical programming formulations	Experiments with uniform data	Hamdi and Loukil (2017)
$F2\|Ch-R, b_j \leqslant L_j = L\|C_{max}, TEC$	ε-constraint method, NSGA II		Liu et al. (2017)
$O2\|\beta\|C_{max}$, where β is:			
$(L_j, a_j = b_j)$	(1.5-ε)-approximation algorithm is NP-hard		Ageev (2018)
$(a_j, L_j \in \{T_1, T_2\}, b_j)$	(1.25-ε)-approximation algorithm is NP-hard		Ageev (2018)

5 Benchmark instances

To the best of our knowledge, there are no published benchmark instances for the coupled task scheduling problem. Several studies, however, generated their own instances to evaluate their proposed algorithms. Often, those instance generation schemes include small- and medium-sized instances, and they may only be applicable to certain special cases. Also, because previously generated instances were not published in the literature, there is a demand for standard test beds; particularly, for evaluation and comparison purposes. Therefore, we propose comprehensive sets of instances for the coupled task scheduling problem, and for both the single-machine and shop environments. Our instance generation scheme incorporates various conditions on the jobs and machines, including number of jobs, values for the processing times and delays, and the machine setting. Our sets also include instances for special cases. Next, we review the previous instance generation schemes and then propose standard benchmark instances for the coupled task scheduling problem.

5.1 Previous instance generation schemes

The first instance generation scheme is due to Shapiro (1980). He generated a number of problem instances by simulating a radar process. The instances set includes a total number of 250 instances with 20 to 500 jobs, where the task processing times are in the range of 10 and 100, and the delay durations in the range of 300 and 1,300. Ahr et al. (2004); Li and Zhao (2007); Brauner et al. (2009); Blazewicz et al. (2012) adopted a similar approach, i.e., taking values from a discrete uniform interval.

The instances in Sherali and Smith (2005) consist of 8 to 14 jobs, with a_j taken from the normal distribution with a mean of 30 time units and a standard deviation of 5 time units, i.e., $N(30, 5)$. Likewise, b_j and L_j are taken from $N(120, 30)$ and $N(600, 100)$, respectively. Condotta and Shakhlevich (2012) defined three types of delay in their instance generation. In the first type, the delays are in a small range, while in the second and third types, the delays take considerably different values. In a set of data in Békési et al. (2014), a_j and b_j take values in the range of 1 and 100, and the values of L_j are generated such that the jobs are perfectly "fitted", i.e., the jobs can be scheduled in such a way that the makespan is at most one unit greater than the trivial lower bound $P_a + P_b$.

Considering the studies in the shop setting, the discrete uniform distribution is used by all the works with a data generation scheme, including Huo et al. (2009); Fondrevelle et al. (2009); Hamdi and Loukil (2017); Amrouche et al. (2017); Meziani et al. (2018a).

As Tables 6 and 7 show, the previous studies generate different instances in order to evaluate their proposed methods. To the best of our knowledge, none of those instances are available. Instead, the instance generation scheme is designed and discussed in those studies, which are not useful for comparing various algorithms and solution methods, because even the same instance generation scheme may lead to different values, e.g., (a_j, L_j, b_j), so different instances. To overcome the limitation, we propose a complete set of benchmark instances for the coupled task scheduling problem, which is publicly available. Section 5.2 explains the procedure and the generated instances for the single-machine problem and Section 5.3 discusses those for the flow shop setting.

Table 6: Characteristics of the data sets proposed in previous studies in the single-machine environment.

Study	Job	(a_j, L_j, b_j)	Number of instances
Shapiro (1980)	$\{20, 50, 100, 200, 500\}$	$([10, 100], [300, 1300], [10, 100])$	250
Ahr et al. (2004)	$\{1000, 2000\}$	$(a_j = a \in [4, 10], L_j = L \in [10, 30], b_j = b \in [2, 5])$	24
Sherali and Smith (2005)	$\{8, 10, 12, 14\}$	$(N(30, 5), N(600, 100), N(120, 30))$	20
Li and Zhao (2007)	$\{50, 100, 200, 500\}$	$([1, 100], [1, 100], [1, 100])$	80
Condotta and Shakhlevich (2012)	$\{20, 30, 50, 100\}$	$([1, 100]$, Three classes, $[1, 100])$	400
		Class 1: $[0.9\frac{\alpha}{2n}\sum(P_a + P_b), 1.1\frac{\alpha}{2n}\sum(P_a + P_b)]$, $\alpha \in \{1, \frac{n}{5}, \frac{n}{5}\}$	
		Class 2: $[1, \alpha]$, $\alpha \in \{5, 10, 25, 50\}$	
		Class 3: $[1, \frac{\alpha}{2n}\sum(P_a + P_b)]$, $\alpha \in \{1, \frac{n}{5}, \frac{n}{5}\}$	
Brauner et al. (2009)	Cyclic	$(5, L_j = L \in [10, 43], 3)$	23
Blazewicz et al. (2012)	$\{2, 3, \ldots, 838\}$	$(1, 4, 1)$	16569
Békési et al. (2014)	$\{8, 9, 10, 11, 12, 20\}$	$([1, 10], [1, 10], [1, 10])$	240
	$\{7, 8, 9, 10, 11, 12\}$	$(1, [1, 10], 1)$	
	$\{7, 8, 9, 10, 11, 12\}$	$(N(30, 5), N(600, 100), N(120, 30))$	
	$\{7, 8, 9, 10, 11, 12\}$	$([1, 100]$, "Fitted" delays, $[1, 100])$	

Table 7: Characteristics of the data sets proposed in previous studies in the flow shop environment.

Study	Job	Machine	$(p_{k,j}, L_{k,j})$	Number of instances
Huo et al. (2009)	$\{20, 40, 60, 80, 100, 120\}$	2	$([1, 100], [1, 100])$	240
Fondrevelle et al. (2009)	$\{14, 16\}$	$\{2, 5\}$	$([20, 50]$, Several classes of delays)	130
Hamdi and Loukil (2017)	$\{5, 10, 15, 20\}$	$\{3, 5, 10\}$	$([20, 50], [0, 100])$	480
Amrouche et al. (2017)	$\{20, 50, 100, 200, 300, 500, 600, 700, 800, 1000\}$	2	Classes 1 to 6: $a_j, b_j, c_j \in ([1, 20], [20, 40]$ or $[40, 60]), L_j = L = 60$	8000
			Class 7: $a_j, b_j, c_j \in [1, 40], L_j = L = 40$	
			Class 8: $a_j, b_j, c_j \in [1, 40], L_j = L = 80$	
Meziani et al. (2018a)	$\{10, 30, 50, 100, 250, 500, 1000\}$	2	Classes 1 to 3: $a_j, b_j, c_j, L_j \in [1, 50]$ or, $\in [50, 100]$, or $\in [100, 150]$	700
			Class 4: $a_j, b_j, L_j \in [100, 150]$ and $c_j \in [1, 50]$	

5.2 The single-machine setting

We include two major factors to generate instances for the single-machine coupled task scheduling problem, namely number of jobs and attributes of the jobs, i.e., values of (a_j, L_j, b_j). The instances with a small number of jobs may be useful for evaluating exact solution methods (see, e.g., Sherali and Smith (2005)). A larger number of jobs are needed for evaluating heuristic algorithms. Therefore, we consider the following set $\{5, 10, 15, 20, 25, 40, 50, 100\}$ for the number of jobs. The previous studies investigate different ranges for the durations of the jobs, e.g., 2 and 10 (Ahr et al., 2004; Békési et al., 2014), as well as 1 and 100 (Li and Zhao, 2007; Condotta and Shakhlevich, 2012). To be inclusive, we investigat jobs with small, medium, and large durations, where the values of (a_j, L_j, b_j) are randomly generated from the discrete uniform distribution in the ranges as follows:

- Small jobs (S): $a_j, b_j \sim U(1, 20), L_j \sim U(10, 80)$
- Medium jobs (M): $a_j, b_j \sim U(1, 50), L_j \sim U(25, 200)$
- Large jobs (L): $a_j, b_j \sim U(1, 100), L_j \sim U(50, 400)$

Given eight values for the number of jobs and three alternatives for their sizes, we have a total of 24 combinations. Generating ten instances for each combination will result in a set with 240 instances. In other words, per each number of jobs, i.e., n, 30 instances are generated, where every ten instances have the same characteristics as discussed above.

We name this set the “General Set”. For illustration purposes, we show the name of an instance in the format `n-x-y-gen`, where `n` denotes the number of jobs, `x` denotes the instance ID (number), ranging from 1 to 10, and `y` denotes the sizes of the jobs that can be `S`, `M`, or `L`, denoting small, medium, or large jobs' attributes as discussed above. The `gen` denotes the General Set. For example, $n = 5$ and small jobs result in a `5-1-S-gen` instance with the following job attributes

$$J_1 = (5, 30, 14), J_2 = (17, 10, 9), J_3 = (15, 52, 6), J_4 = (7, 33, 18), J_5 = (19, 41, 8).$$

Although there is no prior research considering the single-machine coupled task problem with a due date related criterion, we generate a set of due dates to complete the proposed data set. Specifically, we follow the approach proposed by Potts and Van Wassenhove (1982), which is also used in Fondrevelle et al. (2009); Hamdi and Loukil (2017). The method is to generate the due dates in the interval $[Px, Py]$, where P is a lower bound for the problem. We use the bound $P = \sum_j P_a + P_b$, and set $x = 0.05$ and $y = 0.95$.

To better reflect the characteristics of the special cases of the single-machine coupled task scheduling problem, where restrictions on the durations of the tasks and/or delays exist, we generate the second set, i.e., the “Restricted Set”. Because the identical problem in which all the jobs are identical is the extreme special case, we generate the Restricted Set based on the identical problem. The set has 240 instances, where all the jobs of a particular instance are identical, i.e., $(a_j = a, L_j = L, b_j = b) \sim U(\alpha, \beta)$ with the same characteristics and values of α and β as those of the General Set. Our naming convention for this set is `n-x-y-res`, where `n`, `x`, and `y` are as before, and `res` denotes the Restricted Set. For example, `5-1-S-res` may represent an instance with 5 small jobs as follows:

$$J_1 = J_2 = J_3 = J_4 = J_5 = (16, 47, 9).$$

Notice that having the two sets allows us to generate instances for every special case, as well as every restriction on the task/delay durations. For example, an instance with five small jobs for the case $(a_j = a, L_j, b_j)$, where all the initial tasks taking an identical processing time can be generated by using the first values from the Restricted Set, and the second and third values from the General Set. This results in

$$J_1 = (16, 30, 14), J_2 = (16, 10, 9), J_3 = (16, 52, 6), J_4 = (16, 33, 18), J_5 = (16, 41, 8).$$

We use `n-x-y-a` to denote this special case, where `a` stands for identical initial tasks. Benchmark instances for the other special cases can also be generated in the same way. For example, the special case $(a_j = p, L_j, b_j = p), p > 0$, where the initial and completion tasks of every job are equal to p, can be generated from the instances for $(a_j = a, L_j, b_j = b)$ and setting $b_j = a, \forall j$. The special case $a_j \geqslant b_j$ can be generated from the General Set by interchanging the processing times of the tasks of job j with $a_j < b_j$. Similarly, instances for the problem with UET tasks, i.e., $(1, L_j, 1)$, can be generated by using the General Set instances and setting all the task processing times to 1. We generate data sets for all these special cases and make them publicly available for future research. Table 8 shows the naming convention for the instances generated for those special cases. In total, we generate 11 sets, each with 240 instances, i.e., 2,640 problem instances for the single-machine coupled task scheduling problem.

Table 8: Naming convention for instances of the single-machine coupled task problem.

Type	Description
`gen`	General Set
`res`	Restricted Set
`a`	Restricted Set $(a_j = a, L_j, b_j)$
`L`	Restricted Set $(a_j, L_j = L, b_j)$
`b`	Restricted Set $(a_j, L_j, b_j = b)$
`aL`	Restricted Set $(a_j = a, L_j = L, b_j)$
`ab`	Restricted Set $(a_j = a, L_j, b_j = b)$
`Lb`	Restricted Set $(a_j, L_j = L, b_j = b)$
`p`	Restricted Set $(a_j = p, L_j, b_j = p)$
`a>b`	Restricted Set (a_j, L_j, b_j), $a_j \geqslant b_j$
`UET`	Restricted Set $(a_j = 1, L_j, b_j = 1)$

5.3 The flow shop problem

In order to generate benchmark instances for the two-machine flow shop coupled task scheduling problem, we use the instance generation parameters of the single-machine problem (see Section 5.2). In particular, the completion task durations can be considered as the processing times of the tasks on the second machine. Hence, all the sets discussed in Section 5.2 can easily be used for the two-machine flow shop coupled task scheduling problem.

For the m-machine flow shop scheduling problem, however, the numbers of jobs n and machines m need to be chosen. The majority of previous studies investigate problems with up to five machines. An exception is Hamdi and Loukil (2017), who solved instances with up to ten machines. Also, the number of jobs is set to at most 20 in the previous studies. To cover a broader range of instances, we set $n = \{10, 20, 50, 100\}$ and $m = \{5, 10, 20\}$, which results in 12 combinations. For each combination, ten instances are generated, so a total of 120 instances are produced. The tasks' processing times and delay durations are randomly taken from a discrete uniform distribution with the following parameters: $p_{k,j} \sim U(1, 100)$ and $L_{k,j} \sim U(1, 20)$. It should be noted that the values of the delay duration are not from a wide range as in the single-machine case. This is because to reflect the true nature of delays in the flow shop, which are mainly related to the transport of the tasks. We also show the name of an instance in the format `n-m-x`, where `n` denotes the number of jobs, `m` denotes the number of machines, and `x` denotes the instance number, ranging from one to ten. In addition, we generate due dates for the jobs following the same method discussed in the single-machine setting, where the lower bound follows Equation (3), and the same values are used for x and y.

6 Mathematical models

In this section we discuss the mathematical programming formulations for the coupled task scheduling problem documented in the literature. We first discuss the models for the single-machine coupled task scheduling problem and then consider those for the flow shop setting.

6.1 The single-machine models

Let $N = \{1, 2, \ldots, n\}$, indexed by j, and $H = \{1, 2, \ldots, 2n\}$, indexed by h, denote sets of jobs and tasks, respectively, where H_{2j-1} and H_{2j} represent the initial and completion tasks of job j, respectively.

Elshafei et al. (2004) proposed the first mathematical formulation for the single-machine coupled task scheduling problem. For this time-indexed model, they discretized the time horizon into a set Θ of unit time slots. The binary decision variable $x_{j,t}, \forall j, t$, takes the value of 1 if job j starts its processing at time slot $t \in \Theta$, and 0 otherwise. They considered an objective function of minimizing the makespan. Model S1 in the following is their formulation.

Model S1

$$z = \min C_{max} \tag{6}$$

subject to

$$C_{max} \geqslant \sum_{t=1}^{\tau_j} (t + P_j - 1) x_{j,t}, \quad 1 \leqslant j \leqslant n, \tag{7}$$

$$\sum_{t=1}^{\tau_j} x_{j,t} = 1, \quad 1 \leqslant j \leqslant n, \tag{8}$$

$$\sum_{j=1}^{n} \sum_{\substack{v \in s_{2j-1} \\ 1 \leqslant v \leqslant \tau_j}} x_{j,v} + \sum_{j=1}^{n} \sum_{\substack{v \in s_{2j} \\ 1 \leqslant v \leqslant \tau_j}} x_{j,v} \leqslant 1, \quad 1 \leqslant t \leqslant T, \tag{9}$$

$$x_{j,t} \in \{0,1\}, \quad 1 \leqslant j \leqslant n, \quad 1 \leqslant t \leqslant T. \tag{10}$$

The objective function (6) minimizes the makespan, where the makespan is enforced to be at least as large as the finishing time of every job (constraints (7)). Here, $\tau_j = T - P_j + 1$ is the latest time that if a_j is scheduled by τ_j, processing of b_j will finish by time T (end of the time horizon), where $P_j = a_j + L_j + b_j$ is total time ("system time") that job j remains in the system after its initial task starts processing. Constraints (8) ensure that every job starts at exactly one time slot, and constraints (9) indicate that at most one job can occupy each time slot, i.e., no two jobs will overlap in the schedule. In other words, time slot t is occupied by a_j, if the starting time of job j is in $s_{2j-1} \in \{t - a_j + 1, \ldots, t\}$, or it is occupied by b_j, if the starting time of job j is in $s_{2j} \in \{t - P_j + 1, \ldots, t - a_j - L_j\}$. Note that the value of v must always remain between 1 and τ_j and any other value is ignored.

Elshafei et al. (2004) also proposed the weighted version of Model S1, in which the objective is to maximize the total weight of the completed jobs within a time limit $T_{max} \in \Theta$ of the time horizon. Model W1 shows their formulation.

Model W1

$$z = \max \sum_{j=1}^{n} \sum_{t=1}^{\tau_j} w_j x_{j,t} \tag{11}$$

subject to

$$\sum_{t=1}^{\tau_j} x_{j,t} \leqslant 1, \quad 1 \leqslant j \leqslant n, \tag{12}$$

$$\sum_{j=1}^{n} \sum_{\substack{v \in s_{2j-1} \\ 1 \leqslant v \leqslant \tau_j}} x_{j,v} + \sum_{j=1}^{n} \sum_{\substack{v \in s_{2j} \\ 1 \leqslant v \leqslant \tau_j}} x_{j,v} \leqslant 1, \quad 1 \leqslant t \leqslant T_{max}, \tag{13}$$

$$x_{j,t} \in \{0,1\}, \quad 1 \leqslant j \leqslant n, \quad 1 \leqslant t \leqslant T_{max}. \tag{14}$$

The objective function (11) maximizes the total weight of the completed jobs, where w_j is the weight of job j. Recall that this model does not require all the jobs to be scheduled, so constraints (12) reflect this. In other words, the constraints ensure that a job should be started at exactly one time slot t, if it is scheduled within the time limit T_{max}. Note that in Model W1, the value of τ_j is calculated by using the value of T_{max}, i.e., $\tau_j = T_{max} - P_j + 1$, instead of using the value of T.

The numbers of decision variables and constraints in the time-indexed formulations, i.e., Models S1 and W1, depend on the number of time slots. Hence, it is very likely that it grows quickly as the size of the problem increases. Sherali and Smith (2005) attempted to overcome this issue by proposing alternative formulations with continuous time horizon. They considered five possible relative configurations for every pair of jobs j and j':

1. processing of job j is finished before job j' starts;
2. processing of job j' is finished before job j starts;
3. the initial task of job j', i.e., $a_{j'}$, is interleaved between the two tasks of job j (this setting is only possible if $a_{j'} \leqslant L_j$ and $b_j \leqslant L_{j'}$);
4. the initial task of job j, i.e., a_j, is interleaved between the two tasks of job j' (this setting is only possible if $a_j \leqslant L_{j'}$ and $b_{j'} \leqslant L_j$); and,
5. job j is nested between the two tasks of job j' (if $L_{j'} \geqslant P_j$), or job j' is nested between the two tasks of job j (if $L_j \geqslant P_{j'}$; note that at most one of the two cases is possible).

Configurations 1 and 2 are always possible if the time horizon is relatively large, i.e., if $T \geqslant P_j + P_{j'}, \forall j, j' \in N$. According to those five configurations, Sherali and Smith (2005) defined a binary decision variable $y_{j,j',v}$, which takes 1 if jobs j and $j', j < j'$, are scheduled relative to each other based on configuration $v = 1, \ldots, 5$. For a configuration v and jobs j and j', $l_{j,j',v}$ and $r_{j,j',v}$ denote two constants representing the earliest and latest times that job j' can start its processing relative to job j, i.e., $l_{j,j',v} \leqslant s_{j'} - s_j \leqslant r_{j,j',v}$, where s_j is the starting time of job j.

The following example illustrates the calculation of those constants. Consider job j with $a_j = 1, L_j = 4$, and $b_j = 3$ and job j' with $a_{j'} = 1, L_{j'} = 10$, and $b_{j'} = 2$. If the time horizon is equal to 40 time units, configurations 1 and 2 are both possible and we have $l_{j,j',1} = 8, r_{j,j',1} = 27, l_{j,j',2} = -32$, and $r_{j,j',2} = -13$. Configuration 3 is possible as $a_{j'} \leqslant L_j$ and $b_j \leqslant L_{j'}$, so we have $l_{j,j',3} = 1$ and $r_{j,j',3} = 4$. Configuration 4 is possible as $a_j \leqslant L_{j'}$ and $b_{j'} \leqslant L_j$, so $l_{j,j',4} = -10$ and $r_{j,j',4} = -8$. Figure 10 shows te relative positions of jobs j and j' under configuration 4. Lastly, configuration 5 is possible for the nesting of job j between the two tasks of job j', so we have $l_{j,j',5} = -3$ and $r_{j,j',5} = -1$.

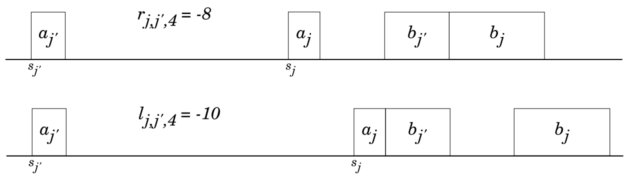

Figure 10: Calculation of $r_{j,j',4}$ and $l_{j,j',4}$ for two jobs j ($a_j = 1, L_j = 4, b_j = 3$) and j' ($a_{j'} = 1, L_{j'} = 10, b_{j'} = 2$)

Following the above definitions, Sherali and Smith (2005) formulated the problem to minimize the makespan into Model S2 as follows:

Model S2

$$z = \min C_{max} \tag{15}$$

subject to

$$C_{max} \geqslant s_j + P_j, \quad 1 \leqslant j \leqslant n, \tag{16}$$

$$\sum_{v=1}^{5} y_{j,j',v} = 1, \quad 1 \leqslant j < j' \leqslant n, \tag{17}$$

$$s_{j'} - s_j \geqslant \sum_{v=1}^{5} l_{j,j',v} y_{j,j',v}, \quad 1 \leqslant j < j' \leqslant n, \tag{18}$$

$$s_{j'} - s_j \leqslant \sum_{v=1}^{5} r_{j,j',v} y_{j,j',v}, \quad 1 \leqslant j < j' \leqslant n, \tag{19}$$

$$s_j \geqslant 0, \quad 1 \leqslant j \leqslant n, \tag{20}$$

$$y_{j,j',v} \in \{0, 1\}, \quad 1 \leqslant j < j' \leqslant n, \quad 1 \leqslant v \leqslant 5. \tag{21}$$

The objective function (15) minimizes the makespan, where the makespan is enforced to be at least as large as the finishing time of every job (constraints (16)). Constraints (17) ensure that exactly one configuration is selected for every pair of jobs. Constraints (18) and (19) ensure that the difference between the starting time of every pair of jobs is within the permissible upper and lower bounds, under their selected configuration. Note that if there is a configuration v that is not possible for two jobs j and j', the relative decision variable $y_{j,j',v}$ is set to zero.

Sherali and Smith (2005) also proposed the weighted version of Model S2. They defined a binary decision variable x_j that is equal to 1 if job j is totally completed, i.e., both of its tasks are processed, before T_{max}. The weighted formulation maximizes the total weight of the completed jobs within the time limit T_{max}. Model W2 is as follows:

Model W2

$$z = \max \sum_{j=1}^{n} w_j x_j \tag{22}$$

subject to

$$s_j + P_j \leqslant UB - x_j(UB - T_{max}), \quad 1 \leqslant j \leqslant n, \tag{23}$$

$$(17) \text{ to } (21) \tag{24}$$

$$x_j \in \{0, 1\}, \quad 1 \leqslant j \leqslant n, \tag{25}$$

where UB is an upper bound for the problem, which can be set to $\sum_j P_j$. Constraints (23) state that if job j is selected to be scheduled, it must finish before the time limit. In Model W2, the values of $l_{j,j',v}$ and $r_{j,j',v}$ are the same as those in Model S3.

Békési et al. (2014) proposed two linear ordering-based formulations of the problem to minimize the makespan. In these formulations, a sequence is an ordered set of tasks, rather than jobs. For any pair of tasks h and h', a binary variable $x_{h,h'}$ is defined, which takes the value of 1 if task h' is started after task h in the sequence. Model S3 shows their first formulation.

Model S3

$$z = \min C_{max} \tag{26}$$

subject to

$$C_{max} \geqslant s_{2j} + b_j, \quad 1 \leqslant j \leqslant n, \tag{27}$$

$$x_{2j-1,2j} = 1, \quad 1 \leqslant j \leqslant n, \tag{28}$$

$$x_{h,h'} + x_{h',h} = 1, \quad 1 \leqslant h < h' \leqslant 2n, \tag{29}$$

$$x_{h,h'} + x_{h',h''} + x_{h'',h} \leqslant 2, \quad \text{for any triple distinct tasks:} \quad h, h', h'' \in H, \tag{30}$$

$$s_{2j} = s_{2j-1} + a_j + L_j, \quad 1 \leqslant j \leqslant n, \tag{31}$$

$$s_{2j} \leqslant UB - b_j, \quad 1 \leqslant j \leqslant n, \tag{32}$$

$$s_{h'} \geqslant s_h + p_h - UB(1 - x_{h,h'}), \quad 1 \leqslant h, h' \leqslant 2n, \quad h \neq h', \tag{33}$$

$$s_h \geqslant 0, \quad 1 \leqslant h \leqslant 2n, \tag{34}$$

$$x_{h,h'} \in \{0, 1\}, \quad 1 \leqslant h, h' \leqslant 2n, \quad h \neq h'. \tag{35}$$

Constraints (28) indicate that the initial task of each job should be scheduled before its completion task. The relative order of any pair of tasks is considered by constraints (29). Constraints (30) are the so-called "3-dicycle inequalities" for any triple distinct tasks. The relation between the starting times of the tasks of a job is defined by constraints (31), where an upper bound on the starting times of the completion tasks is set by constraints (32). Constraints (33) relate the starting time variables to the linear ordering variables, where p_h is the processing time of task h. Specifically, the constraints ensure that if task h' is scheduled after task h, the starting time of task h' must be at least as large as the finishing time of task h.

Békési et al. (2014) proposed a second formulation as an alternative for Model S3. The model does not use the starting time variables. The model includes variables y_h, which are the idle times of the machine after processing task h, and positive constants C_h, which are upper bounds on the values of y_h. A Knapsack constraint is used in order to express that the sum of the processing times of tasks that are executed between the two tasks of job j must not exceed its delay duration L_j. Because such a constraint is initially stated non-linear, additional variables $y_{j,h}, 1 \leqslant j \leqslant n, 1 \leqslant h \leqslant 2n, h \notin \{2j-1, 2j\}$ are added for linearization. Model S4 is the formulation as follows:

Model S4

$$z = \min \sum_{j=1}^{n} y_{2j-1} + y_{2j} \tag{36}$$

subject to

$$y_{2j-1} + \sum_{h \notin \{2j-1,2j\}} (p_h(x_{h,2j} - x_{h,2j-1}) + y_{j,h}) = L_j, \quad 1 \leqslant j \leqslant n, \tag{37}$$

$$y_{j,h} \leqslant y_h, \quad 1 \leqslant j \leqslant n, \quad 1 \leqslant h \leqslant 2n, \quad h \notin \{2j-1, 2j\}, \tag{38}$$

$$y_{j,h} \leqslant C_{j,h}(x_{h,2j} - x_{h,2j-1}), \quad 1 \leqslant j \leqslant n, \quad 1 \leqslant h \leqslant 2n, \quad h \notin \{2j-1, 2j\}, \tag{39}$$

$$y_{j,h} \geqslant y_h - C_j(x_{2j,h} + x_{h,2j-1}), \quad 1 \leqslant j \leqslant n, \quad 1 \leqslant h \leqslant 2n, \quad h \notin \{2j-1, 2j\}, \tag{40}$$

$$(28) \text{ to } (30) \tag{41}$$

$$y_j \geqslant 0, \quad 1 \leqslant j \leqslant n, \tag{42}$$

$$y_{j,h} \geqslant 0, \quad 1 \leqslant j \leqslant n, \quad 1 \leqslant h \leqslant 2n, \quad h \notin \{2j-1, 2j\}. \tag{43}$$

The objective function (36) minimizes the total idle time of the machine, which is equivalent to minimizing the makespan. The Knapsack constraints are presented in (37), indicating that the delay duration of each job consists of the tasks that are processed between its initial and completion tasks, plus the idle time between them. Constraints (38) to (40) are used to linearize the Knapsack constraints. Here, the constants $C_{j,h} = \max\{0, L_j - p_h\}$ are upper bounds on the values of $y_{j,h}$. In addition, $C_h = L_j$, if h is the first task of job j, or $C_h = \max_j\{C_{j,h} | h \notin \{2j-1, 2j\}\}$, if h is the second task of a job.

6.2 The flow shop models

Hamdi and Loukil (2017) proposed three mathematical formulations for the flow shop coupled task scheduling problem. The objective function of the models is to minimize the total earliness and tardiness. The first model, which we denote by F1, is a completion time-based formulation. It uses the decision variable $x_{i,j}$, which takes the value of 1 if job j is assigned to the sequence position i, and 0 otherwise. Also, the non-negative variable $C_{i,k}$ denotes the completion time of job in the sequence position i on machine k. Model F1 shows this formulation as follows:

Model F1

$$z = \min \sum_{i=1}^{n} (E_i + T_i) \tag{44}$$

subject to

$$\sum_{i=1}^{n} x_{i,j} = 1, \quad 1 \leqslant j \leqslant n, \tag{45}$$

$$\sum_{j=1}^{n} x_{i,j} = 1, \quad 1 \leqslant i \leqslant n, \tag{46}$$

$$T_i \geqslant C_{i,m} - \sum_{j=1}^{n} d_j x_{i,j}, \quad 1 \leqslant i \leqslant n, \tag{47}$$

$$E_i \geqslant \sum_{j=1}^{n} d_j x_{i,j} - C_{i,m}, \quad 1 \leqslant i \leqslant n, \tag{48}$$

$$C_{i,k+1} = C_{i,k} + \sum_{j=1}^{n} (p_{j,k+1} + L_{j,k}) x_{i,j}, \quad 1 \leqslant i \leqslant n, \quad 1 \leqslant k \leqslant m-1, \tag{49}$$

$$C_{i+1,k} \geqslant C_{i,k} + \sum_{j=1}^{n} p_{j,k} x_{i+1,j}, \quad 1 \leqslant i \leqslant n-1, \quad 1 \leqslant k \leqslant m, \tag{50}$$

$$C_{i,k} \geqslant 0, \quad 1 \leqslant i \leqslant n, \quad 1 \leqslant k \leqslant m, \tag{51}$$

$$x_{i,j} \in \{0,1\}, \quad 1 \leqslant i,j \leqslant n, \tag{52}$$

$$E_i, T_i \geqslant 0, \quad 1 \leqslant i \leqslant n. \tag{53}$$

The objective function (44) minimizes the total earliness and tardiness. The assignment constraints (45) and (46) ensure that each job is assigned to exactly one sequence position and each sequence position is assigned to exactly one job. Constraints (47) and (48) define the tardiness and earliness of the job in the sequence position i. Constraints (49) define the completion time of each job on consecutive machines. Likewise, constraints (50) calculate the completion times of any pair of consecutive jobs on every machine. We notice that Hamdi and Loukil (2017) missed a constraint in Model F1, i.e., they did not explicitly define the completion time of the first job on the first machine. In particular, the omission of such a constraint may result in starting the first job on the first machine before the time horizon, i.e., an infeasible solution. To avoid this, we add the following constraint to Model F1.

$$C_{1,1} \geqslant \sum_{j=1}^{n} p_{j,1} x_{1,j}. \tag{54}$$

We express constraint (54) in the form of greater than or equal to because the objective function of Model F3, i.e., the total earliness and tardiness is a non-regular one, and forced idle times may therefore improve the objective. In case of a regular objective function, e.g., minimization of the makespan, forced idle times will not be beneficial, so it suffices to express constraint (54) as $C_{1,1} = \sum_{j=1}^{n} p_{j,1} x_{1,j}$.

Their second model is an idle time-based formulation, in which the variable $I_{i,k}$ is the idle time of machine k after processing the job in the sequence position i. We notice that their formulation does not generate a valid solution. We investigate their formulation and observe two flaws. First, the calculation of jobs' earliness needs fixing. Second, by defining variable $I_{i,k}$ as the idle time of machine k "after" processing the job in the sequence position i, there is no possibility to have idle time before the processing of the first job on the first machine. Note that because the objective function is non-regular, it might be beneficial to have idle time before the processing of the first job on the first machine. Consequently, we define variable $I_{i,k}$ as the idle time of machine k "before" processing the job in the sequence position i. We also correct the earliness calculation. The corrected formulation, denoted as Model F2, is as follows:

Model F2

$$\text{(44) to (46)} \tag{55}$$

$$\sum_{j=1}^{n} (p_{j,k} + L_{j,k}) x_{1,j} + I_{1,k} = I_{1,k+1}, \quad 1 \leqslant k \leqslant m-1, \tag{56}$$

$$\begin{aligned}&\sum_{j=1}^{n} (p_{j,k+1} + L_{j,k}) x_{i,j} + I_{i+1,k+1} \\ &\quad = \sum_{j=1}^{n} (p_{j,k} + L_{j,k}) x_{i+1,j} + I_{i+1,k}, \quad 1 \leqslant i \leqslant n-1, \quad 1 \leqslant k \leqslant m-1,\end{aligned} \tag{57}$$

$$T_i \geqslant \sum_{s=1}^{i} \sum_{j=1}^{n} p_{j,m} x_{s,j} + \sum_{s=1}^{i} I_{s,m} - \sum_{j=1}^{n} d_j x_{i,j}, \quad 1 \leqslant i \leqslant n, \tag{58}$$

$$E_i \geqslant \sum_{j=1}^{n} d_j x_{i,j} - \sum_{s=1}^{i} \sum_{j=1}^{n} p_{j,m} x_{s,j} - \sum_{s=1}^{i} I_{s,m}, \quad 1 \leqslant i \leqslant n, \tag{59}$$

$$I_{i,k} \geqslant 0, \quad 1 \leqslant i \leqslant n, \quad 1 \leqslant k \leqslant m, \tag{60}$$

$$\text{(52) and (53).} \tag{61}$$

Constraints (56) model the idle time before the job in the first position. Similarly, constraints (57) define the idle time between the remaining jobs on all the machines. In particular, they relate the processing times, delays, and idle times of every two consecutive jobs on any pair of consecutive machines. Constraints (58) and (59) define the tardiness and earliness, respectively.

The third model proposed by Hamdi and Loukil (2017) is a starting time-based formulation, which uses the variable $s_{i,k}$ to express the starting time of the job in position i on machine k. Model F3 in the following shows this formulation.

Model F3

$$\text{(44) to (46)} \tag{62}$$

$$T_i \geqslant s_{i,m} + \sum_{j=1}^{n} p_{j,m} x_{i,j} - \sum_{j=1}^{n} d_j x_{i,j}, \quad 1 \leqslant i \leqslant n, \tag{63}$$

$$E_i \geqslant \sum_{j=1}^{n} d_j x_{i,j} - s_{i,m} - \sum_{j=1}^{n} p_{j,m} x_{i,j}, \quad 1 \leqslant i \leqslant n, \tag{64}$$

$$s_{i,k+1} = s_{i,k} + \sum_{j=1}^{n} (p_{j,k} + L_{j,k}) x_{i,j}, \quad 1 \leqslant i \leqslant n, \quad 1 \leqslant k \leqslant m-1, \tag{65}$$

$$s_{i+1,k} \geqslant s_{i,k} + \sum_{j=1}^{n} p_{j,k} x_{i,j}, \quad 1 \leqslant i \leqslant n-1, \quad 1 \leqslant k \leqslant m, \tag{66}$$

$$s_{i,k} \geqslant 0, \quad 1 \leqslant i \leqslant n, \quad 1 \leqslant k \leqslant m, \tag{67}$$

$$\text{(52) and (53).} \tag{68}$$

Constraints (63) and (64) define the tardiness and earliness of jobs. Constraints (65) relate the starting times of a job on every pair of consecutive machines. Finally, constraints (66) relate the starting times of any pair of consecutive jobs on a machine. Their original model also includes the constraint $s_{1,1} \geqslant 0$. This constraint is redundant because it has already been implied by constraints (67). Therefore, we do not include it in Model F3.

Also, we provide an additional formulation for the flow shop coupled task scheduling problem, which Arabameri and Salmasi (2013) originally proposed for the no-wait flow shop scheduling problem. We choose this model because it is relatively new and uses the sequential ordering variables (the first three have positional variables).

The model proposed by Arabameri and Salmasi (2013) includes a binary variable $x_{j,j'}$, which takes 1 if job j' is placed immediately after job j in a sequence, and 0 otherwise. Two dummy jobs are considered, where their processing times and delays on all the machines are equal to zero. The due date of the first dummy job is also set to zero so as to assign it to the first position in a sequence, and that of the second dummy job is set to a large positive number in order to assign it to the last position in the sequence. Model F4 is as follows:.

Model F4

$$z = \min \sum_{j=1}^{n'-1} (E_j + T_j) \tag{69}$$

subject to

$$\sum_{\substack{j=1 \\ j \notin j'}}^{n'-1} x_{j,j'} = 1, \quad 2 \leqslant j' \leqslant n', \tag{70}$$

$$\sum_{\substack{j'=2 \\ j' \notin j}}^{n'} x_{j,j'} = 1, \quad 1 \leqslant j \leqslant n' - 1, \tag{71}$$

$$C_{j',k} + M(1 - x_{j,j'}) \geqslant C_{j,k} + p_{j,k}, \quad 1 \leqslant j, j' \leqslant n', \quad j \notin j', \quad 1 \leqslant k \leqslant m, \tag{72}$$

$$C_{j,k} = C_{j,k-1} + p_{j,k} + L_{j,k-1}, \quad 1 \leqslant j \leqslant n', \quad 2 \leqslant k \leqslant m, \tag{73}$$

$$T_j \geqslant C_{j,m} - d_j, \quad 1 \leqslant j \leqslant n', \tag{74}$$

$$E_j \geqslant d_j - C_{j,m}, \quad 1 \leqslant j \leqslant n', \tag{75}$$

$$C_{j,k} \geqslant 0, \quad 1 \leqslant j \leqslant n', \quad 1 \leqslant k \leqslant m, \tag{76}$$

$$x_{j,j'} \in \{0, 1\}, \quad 1 \leqslant j, j' \leqslant n', \quad j \notin j', \tag{77}$$

$$E_j, T_j \geqslant 0, \quad 1 \leqslant j \leqslant n'. \tag{78}$$

The objective function (69) minimizes the total earliness and tardiness of jobs 1 to $n' - 1$, where $n' = n + 2$. The last job is not included as it is the second dummy job. Constraints (70) and (71) ensure that all the jobs are sequenced exactly once. Constraints (72) and (73) relate the completion time variables to the sequential ordering variables. Specifically, they set the completion times of every two consecutive jobs on every machine, and every job on any pair of consecutive machines. Here $C_{j,k}$ is the completion time of job j on machine k and M is a sufficiently large constant. Constraints (74) and (75) define the tardiness and earliness of the jobs.

7 Performance evaluation of models

In this section we evaluate the performance of all ten models discussed in Section 6. For this purpose, we carried out comprehensive computational experiments. Recall that the mathematical models can be categorized into three groups: (1) single-machine models to minimize the makespan, i.e., Models S1 to S4; (2) single-machine models to minimize the weighted sum of the completed jobs, i.e., Models W1 and W2; and (3) flow shop models to minimize the total earliness-tardiness, i.e., Models F1 to F4. Table 9 summarizes the size of each model, i.e., number of decision variables and constraints. In the table, n and m denote the numbers of jobs and machines, respectively, and T denotes the number of discretized time units.

Table 9: Numbers of decision variables and constraints in the models.

Model	Number of binary variables	Number of continuous variables	Number of constraints
S1	nT	1	$2n + T$
S2	$(5n^2 - 5n)/2$	$n + 1$	$(3n^2 - n)/2$
S3	$4n^2 - 2n$	$n + 1$	$8n^3 - 6n^2 + 5n$
S4	$4n^2 - 2n$	$2n^2$	$8n^3 - 4n^2 - n$
W1	nT	-	$n + T_{max}$
W2	$(5n^2 - 3n)/2$	n	$(3n^2 - n)/2$
F1	n^2	$nm + 2n$	$2nm + 3nm + 1$
F2	n^2	$nm + 2n$	$nm + 3n$
F3	n^2	$nm + 2n$	$2nm + 3nm + 1$
F4	$(n + 2)^2 - n - 2$	$(n + 2)m + 2(n + 2)$	$(n + 2)^2 m + 3(n + 2) - 2$

For the first and second groups, which are models for the single-machine coupled task scheduling problem, we solved 240 instances of General Set (see Section 5.2). For the weighted models, we randomly generated the job weights from $\sim U(1, 10)$. We set the parameter T_{max} to $\frac{1}{3} \sum_j P_j$, where $P_j = a_j + L_j + b_j$. For the third group, i.e., the flow shop coupled task scheduling models, we solved 120 benchmark instances generated in Section 5.3. We used the solver Gurobi version 8.0.0 (Gurobi Optimization, 2018) to solve the models. We coded all the models in Python version 2.7. We

Table 10: Comparison of the performance of the studied mathematical models.

Model	Feas	Best	Opt	Gap (%)	Time (sec)
S1	**210**	73	34	30.9	788.2
S2	126	85	**61**	4.2	**674.8**
S3	192	**148**	60	**3.3**	677.3
S4	166	68	58	15.5	751.7
W1	**240**	**199**	102	**0.4**	578.4
W2	205	153	**113**	1.1	**497.5**
F1	106	32	30	5.5	675.5
F2	107	47	**36**	5.1	**645.0**
F3	109	32	30	5.6	675.5
F4	**120**	**106**	30	**0.6**	742.7

performed all the computational experiments on a PC with an Intel® Core$^{\text{TM}}$ i5-7500 CPU clocked at 3.40GHz with 8GB of memory under the Linux Ubuntu 16.04 operating system. The machine has four threads and we ran Gurobi in the parallel mode, i.e., we used all the four threads. We only changed one Gurobi parameter, i.e., the time limit, for which we changed its default value to 900 seconds. Therefore, this was the stopping criterion for the solver. For the remaining parameters of Gurobi, we used their default values.

Table 10 summarizes the results of the computational experiments. We use five criteria to evaluate each model. These include "Feas", which shows the number of generated feasible solutions by the model within the time limit; "Best", which is the number of best solutions obtained by the model; "Opt", i.e., the number of optimal solutions delivered by the model; and "Gap (in %)", which is calculated as $\frac{z-z^*}{z^*} \times 100$, where z is the objective function value obtained by the model and z^* is the best objective function value among all the models, i.e., the best available solution, and is calculated over the number of feasible solutions for the model. The last metric "Time (sec)" represents the average computing time in seconds for the model. We highlight the criterion with the best value across the models (per group).

Among the single-machine models to minimize the makespan, i.e., Models S1 to S4, the time-indexed model, i.e., Model S1, generates the largest number of feasible solutions. In addition, as Table 12 shows, Model S1 is the only model that produces feasible solutions for instances with 100 jobs. Despite this, Model S1 is not the outperforming model in terms of solution quality because it only delivers the best solution for 73 (almost 30%) and optimal for 34 (less than 15%) of instances. The model's gap (from the best solution) is also the highest among all the models. Notice that a larger value of gap implies that the solution is further from the best available one.

Model S2 delivers feasible solution for slightly more than half of instances (52.5%). Nevertheless, it obtains the highest number of optimal solutions. Also, its gap is the second best, acknowledging the quality of its solutions. Table 12 details this and shows that the majority of high-quality solutions obtained by Model S2 are for instances with up to 20 jobs. Model S2 was originally proposed to overcome the exponential growth in the numbers of variables and constraints in the time-indexed model. However, it is not difficult to observe that it is not able to deliver feasible solutions for instances with 40 jobs and more.

The linear ordering model, i.e., Model S3, obtains the highest number of best solutions. It also delivers the optimal solutions for a quarter of the instances. The quality of the delivered solutions is very promising because the gap is lowest among all the models. The model generates feasible solutions for 80% of the instances. The details presented in Table 12 indicate that this model has the capability of Models S1 and S2 since it generates quality solutions for a broad range of the instances. Model S4, which is an alternative for Model S3, under-performs Model S3 across all the metrics.

Figure 11 depicts the details of the computing times for Models S1 to S4. The instances are sorted in non-decreasing order of the computing times of Model S2, only for the sake of better illustration. According to the figure, Models S2 and S3 are able to solve more instances in a short time because their computing times do not quickly increase as those for Models S1 and S4.

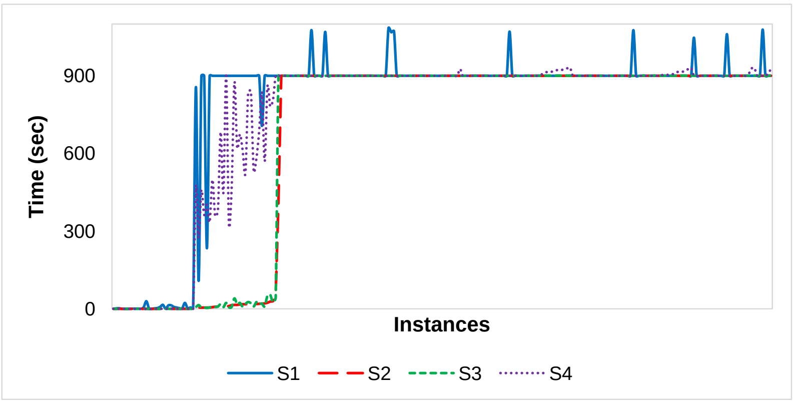

Figure 11: Detailed computing times for single-machine models to minimize the makespan.

Between the weighted models, Model W1 generates feasible solutions for all the instances, while Model W2 produces feasible solutions for 205 (almost 86%) instances. Model W1's better performance is further recognized by its larger number of best solutions and lower gap values. Particularly, the average gap for Model W1 is less than half of that for Model W2. Regarding the number of optimal solutions, however, Model W2 obtains more optimal solutions, and in the order of 11 more instances. Based on the detailed results presented in Table 13, we draw the conclusion that Model W2 is a better choice to solve the small instances, due to its shorter computing times and greater numbers of optimal solutions. On the contrary, Model W1 is the best choice when dealing with large instances, due to its competitive gap and number of best solutions obtained. Figure 12 depicts the details of the computing times for Models W1 and W2. The instances are sorted in non-decreasing order of the computing times for Model W2.

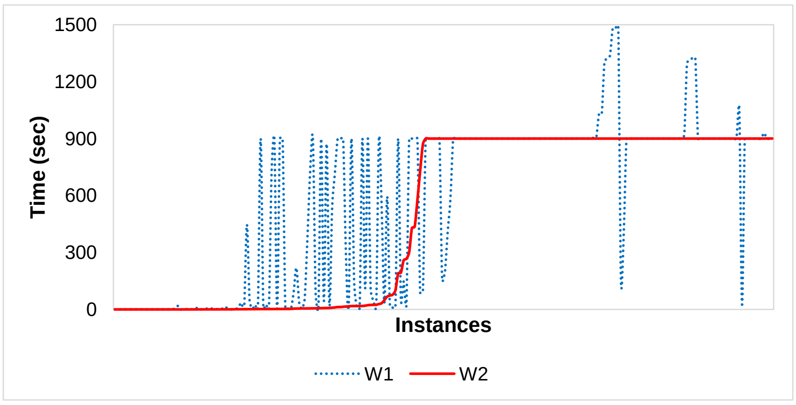

Figure 12: Detailed computing times for single-machine models to minimize the weighted sum of the completed jobs.

Among the flow shop models, i.e., Models F1 to F4, the adapted sequential ordering model, i.e., Model F4, performs significantly better than the other models. In particular, it delivers feasible solutions for all the instances, with 106 (nearly 88%) of which being the best available solutions. As a result, the gap value of this model is small, i.e., 0.6%. Models F1 and F3 behave very closely, which is not surprising since they are very similar. Model F2, i.e., the idle time-based model, is slightly better than Models F1 and F3 because it has a lower average gap, as well as greater number of best and optimal solutions. Model F2 also reports the largest number of optimal solutions among all the models. The average computing times of Models F1 to F3 are less than that of Model F4; particularly, Model F2 has the shortest average time. To conclude, the detailed results presented in Table 14 reveal that Model F2 is the best option to solve instances with up to 10 jobs. However, Model F4 is the top performing choice for large instances. Figure 13 details the computation times for four models of flow shop coupled task scheduling problem. Note that the instances are sorted in a non-decreasing order of computation time of Model F2.

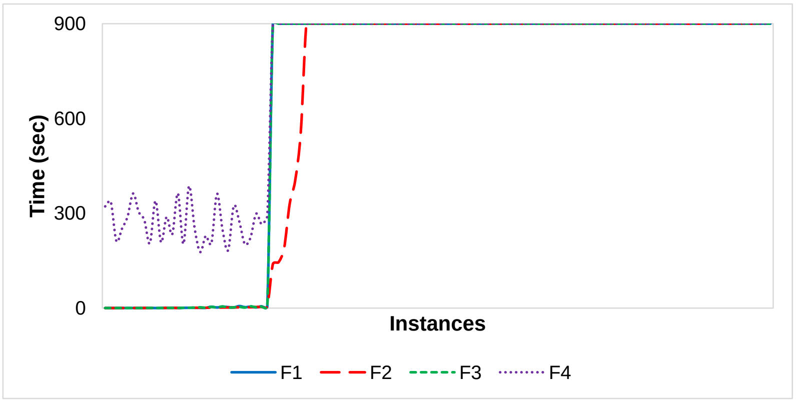

Figure 13: Detailed computing times for flow shop models to minimize the total earliness-tardiness.

We further investigate the quality of the Linear Programming (LP) relaxation objective function associated with the models. For this, we calculate the "LP gap (in %)" as $\frac{z^*-z^{LP}}{z^*} \times 100$, which represents the the percentage gap between the LP relaxation objective function z^{LP} (generated by the solver) and z^* (the best available solution). Table 11 summarizes the results, where N_{LP} is the number of instances that the model is able to produce the LP solution. Models S2 and S4, among the single-machine models to minimize the makespan, generate the LP relaxation solution for 207 instances, and is greater than that of the other two models. Also, we observe that the quality of the bounds generated by Model S4 is far better than that of Model S2. Considering the weighted models, Model W2 delivers the LP relaxation solution for a slightly greater number of the instances; however, the quality of the bounds of W1 is better. All the flow shop models

Table 11: Comparison of the LP relaxation objective function generated by the studied mathematical models.

Model	LP gap (%)	N_{LP}
S1	40.7	155
S2	69.0	207
S3	67.4	197
S4	17.3	207
W1	3.0	211
W2	4.7	220
F1	26.2	120
F2	26.2	120
F3	26.2	120
F4	67.0	120

generate LP bounds for all the instances. In addition, the quality of the LP bounds for the three models of F1 to F3 are identical, and is far better than that of Model F4.

Table 12: Detailed performance of the single-machine makespan models (a "-" denotes that the model cannot produce an outcome within the time limit).

n	S1					S2					S3					S4				
	Feas	Best	Opt	Gap (%)	Time (sec)	Feas	Best	Opt	Gap (%)	Time (sec)	Feas	Best	Opt	Gap (%)	Time (sec)	Feas	Best	Opt	Gap (%)	Time (sec)
5	30	30	30	0.0	5.0	30	30	30	0.0	0.0	30	30	30	0.0	0.0	30	30	30	0.0	0.2
10	30	5	4	4.7	843.5	30	30	30	0.0	15.0	30	30	30	0.0	17.8	30	30	28	0.0	596.8
15	30	0	0	12.1	900.1	30	10	1	1.0	883.1	30	21	0	0.2	900.0	30	0	0	7.5	900.0
20	30	1	0	28.6	900.1	27	8	0	8.9	900.0	30	22	0	0.4	900.1	30	0	0	13.4	900.0
25	30	7	0	60.2	900.1	9	7	0	28.8	900.0	30	16	0	1.8	900.2	25	0	0	48.8	900.0
40	30	11	0	70.6	956.9	0	0	0	-	900.0	26	15	0	21.3	900.0	16	4	0	42.2	900.0
50	20	9	0	60.1	900.1	0	0	0	-	900.0	16	14	0	0.4	900.1	5	4	0	10.8	900.0
100	10	10	0	0.0	900.1	0	0	0	-	900.0	0	0	0	-	900.6	0	0	0	-	916.8

Table 13: Detailed performance of the single-machine weighted models (a "-" denotes that the model cannot produce an outcome within the time limit).

n	W1					W2				
	Feas	Best	Opt	Gap (%)	Time (sec)	Feas	Best	Opt	Gap (%)	Time (sec)
5	30	30	30	0.0	1.5	30	30	30	0.0	0.0
10	30	23	20	0.8	363.8	30	30	30	0.0	6.1
15	30	25	20	0.6	390.7	30	30	22	0.0	274.7
20	30	19	12	0.8	589.2	30	27	14	0.1	517.0
25	30	21	8	0.6	690.1	30	19	9	1.1	651.3
40	30	24	6	0.5	750.5	30	12	5	3.0	802.6
50	30	27	6	0.3	785.1	25	5	3	4.2	828.3
100	30	30	0	0.0	1056.3	0	0	0	-	900.1

Table 14: Detailed performance of the flow shop models.

n	F1					F2					F3					F4				
	Feas	Best	Opt	Gap (%)	Time (sec)	Feas	Best	Opt	Gap (%)	Time (sec)	Feas	Best	Opt	Gap (%)	Time (sec)	Feas	Best	Opt	Gap (%)	Time (sec)
10	30	30	30	0.0	1.8	30	30	30	0.0	1.3	30	30	30	0.0	1.6	30	30	30	0.0	270.6
20	30	1	0	1.4	900.0	30	7	6	0.9	778.8	30	2	0	1.2	900.0	30	27	0	0.0	900.0
50	26	0	0	10.4	900.1	30	0	0	13.7	900.0	29	0	0	9.2	900.1	30	30	0	0.0	900.0
100	20	1	0	13.7	900.1	17	10	0	6.2	900.0	20	0	0	15.2	900.1	30	19	0	2.3	900.1

8 Discussion and concluding remarks

It has been about 40 years since the coupled task scheduling problem was introduced to the scheduling community. The early years witnessed only a few studies on this topic, and the majority of works and interests have therefore been developed in the last few years. One factor contributing to this is that several real-world applications can be modelled as the coupled task scheduling problem. Therefore, we believe that the present review provides a timely survey of previous studies, which can be a valuable resource in guiding and directing future research on the topic. In particular, we present benchmark instances and carry out comprehensive evaluation of the available mathematical models for the problem, so providing fundamental background for research into algorithm development and computational performance. The fact that the exact delay is a generalization of the well-known "no-wait" scheduling condition highlights the theoretical and practical importance of studying the coupled task scheduling problems. In this section we summarize research on the coupled task scheduling problem, which we discussed in detail in the previous section, and highlight the main open problems, as well as potential areas for future research.

8.1 Solution methods

In the single-machine setting, the interleaving and nesting heuristics of Shapiro (1980) are of importance for developing solution methods. Two Tabu Search (TS) algorithms were proposed by Li and Zhao (2007); Condotta and Shakhlevich (2012), though no detailed comparison was reported. Two notable exact algorithms of B&B (Békési et al., 2014) and dynamic programming (Ahr et al., 2004) were proposed; the latter is for the case of the identical problem. In the domain of approximation algorithms, simple dispatching rules including SPT and LPT have been widely used in order to derive approximation ratios, e.g., by Ageev and Kononov (2007); Ageev and Ivanov (2016).

Concerning the flow shop setting, approximation algorithms have been proposed that utilize similar dispatching rules or algorithms for the single-machine setting. For the objective function of minimizing the total completion time, a TS and a SA metaheuristics were proposed by Huo et al. (2009), and the SA was shown to slightly perform better. For the objective function of minimizing the maximum lateness, a B&B algorithm was proposed in Fondrevelle et al. (2009). Three metaheuristics of Particle Swarm Optimization (PSO), SA, and a hybrid (SA+PSO) were proposed for the two-machine flow shop, where the coupled tasks only exist on the first machine. They showed that the hybrid algorithm outperforms the others. For the only bi-objective study in this domain (minimizing the makespan and the total energy consumption), Liu et al. (2017) proposed the ε-constraint and a Non-dominated Sorting Genetic Algorithm II (NSGA II) algorithms, but provided no computational experiments.

8.2 Future research directions

We believe the following areas are worth further investigation, so we suggest them as future research directions in this domain.

Computational complexity under different objective functions. Section 3.1.1 details the complexity of the single-machine coupled task scheduling problem to minimize the makespan. Here, an important problem is the identical case, i.e., (a, L, b). Ahr et al. (2004); Baptiste (2010) developed several preliminary results for this case, though its complexity still remains open.

In addition, 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. Investigating the computational complexity of the problem with different objective functions is valuable because of practical applications of those performance criteria. Moreover, the multi-objective coupled task problem is an interesting topic for future research, especially due to the involvement of (conflicting) objectives in real-world applications.

Solution methods. Apart from simple dispatching rules, heuristics, and approximation algorithms, there are very few advanced and effective solution methods for both the single-machine and flow shop coupled task scheduling problems. In view of the NP-hardness of the majority of the cases, developing effective and efficient solution methods is highly desirable.

Scheduling environment. Research in the context of the single-machine problem is more established than that in the shop setting. While in the shop setting, the flow shop coupled task scheduling problem has received fairly good attention, research in other shop environments, e.g., the job and flexible shop settings, needs further development. We observe that investigation into the open shop setting is evolving; however, it is still in the very preliminary stage. It should be noted that the parallel-machine setting is of significant importance because it may be applied to model the more complex flexible shop scheduling problem.

Acknowledgments

Mostafa Khatami is the recipient of UTS International Research Scholarship (IRS) and UTS President's Scholarship (UTSP). Dr Amir Salehipour is the recipient of an Australian Research Council Discovery Early Career Researcher Award (DECRA) funded by the Australian Government. Cheng was supported in part by The Hong Kong Polytechnic University under the Fung Yiu King - Wing Hang Bank Endowed Professorship in Business Administration.

References

Ageev, A. A. (2018). "Inapproximately lower bounds for open shop problems with exact delays". *Approximation and Online Algorithms*. Springer International Publishing AG, 45–55.

Ageev, A. A. and Baburin, A. E. (2007). "Approximation algorithms for UET scheduling problems with exact delays". *Operations Research Letters* 35(4), 533 –540.

Ageev, A. A. and Ivanov, M. (2016). "Approximating coupled-task scheduling problems with equal exact delays". *Discrete Optimization and Operations Research*. Springer International Publishing: Cham, 259–271.

Ageev, A. A. and Kononov, A. V. (2007). "Approximation algorithms for scheduling problems with exact delays". *Approximation and Online Algorithms*. Springer Berlin Heidelberg.

Ahr, D., Békési, J., Galambos, G., Oswald, M., and Reinelt, G. (2004). "An exact algorithm for scheduling identical coupled tasks". *Mathematical Methods of Operations Research* 59(2), 193–203.

Allahverdi, A. (2016). "A survey of scheduling problems with no-wait in process". *European Journal of Operational Research* 255(3), 665 –686.

Amrouche, K. and Boudhar, M. (2016). "Two machines flow shop with reentrance and exact time lag". *RAIRO-Operations Research* 50(2), 223–232.

Amrouche, K., Boudhar, M., Bendraouche, M., and Yalaoui, F. (2017). "Chain-reentrant shop with an exact time lag: new results". *International Journal of Production Research* 55(1), 285–295.

Arabameri, S. and Salmasi, N. (2013). "Minimization of weighted earliness and tardiness for no-wait sequence-dependent setup times flowshop scheduling problem". *Computers & Industrial Engineering* 64(4), 902 –916.

Baptiste, P. (2010). "A note on scheduling identical coupled tasks in logarithmic time". *Discrete Applied Mathematics* 158(5), 583 –587.

Békési, J., Galambos, G., Oswald, M., and Reinelt, G. (2009). "Improved analysis of an algorithm for the coupled task problem with UET jobs". *Operations Research Letters* 37(2), 93 –96.

Békési, J., Galambos, G., Jung, M. N., Oswald, M., and Reinelt, G. (2014). "A branch-and-bound algorithm for the coupled task problem". *Mathematical Methods of Operations Research* 80(1), 47–81.

Bessy, S. and Giroudeau, R. (2018). "Parameterized complexity of a coupled-task scheduling problem". *Journal of Scheduling*.

Blazewicz, J., Ecker, K., Kis, T., Potts, C. N., Tanas, M., and Whitehead, J. (2010). "Scheduling of coupled tasks with unit processing times". *Journal of Scheduling* 13(5), 453–461.

Blazewicz, J., Pawlak, G., Tanas, M., and Wojciechowicz, W. (2012). "New algorithms for coupled tasks scheduling - a survey". *RAIRO-Operations Research* 46(4), 335–353.

Brauner, N., Finke, G., Lehoux-Lebacque, V., Potts, C., and Whitehead, J. (2009). "Scheduling of coupled tasks and one-machine no-wait robotic cells". *Computers & Operations Research* 36(2), 301 –307.

Chu, C. and Proth, J. (1996). "Single machine scheduling with chain: Structured precedence constraints and separation time windows". *IEEE Transactions on Robotics and Automation* 12(6), 835–844.

Condotta, A. and Shakhlevich, N. (2012). "Scheduling coupled-operation jobs with exact time-lags". *Discrete Applied Mathematics* 160(16), 2370 –2388.

Condotta, A. and Shakhlevich, N. (2014). "Scheduling patient appointments via multilevel template: A case study in chemotherapy". *Operations Research for Health Care* 3(3), 129 –144.

Darties, B., Giroudeau, R., König, J.-C., and Simonin, G. (2016). "Some complexity and approximation results for coupled-tasks scheduling problem according to topology". *RAIRO-Operations Research* 50(4), 781–795.

Dell'Amico, M. (1996). "Shop problems with two machines and time lags". *Operations Research* 44(5), 777–787.

Ecker, K. and Tanaś, M. (2012). "Complexity of scheduling of coupled tasks with chains precedence constraints and any constant length of gap". *Journal of the Operational Research Society* 63(4), 524–529.

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

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

Fondrevelle, J., Oulamara, A., Portmann, M.-C., and Allahverdi, A. (2009). "Permutation flow shops with exact time lags to minimise maximum lateness". *International Journal of Production Research* 47(23), 6759–6775.

Giaro, K. (2001). “NP-hardness of compact scheduling in simplified open and flow shops”. *European Journal of Operational Research* 130(1), 90 –98.

Gilmore, P. C. and Gomory, R. E. (1964). “Sequencing a one state-variable machine: A solvable case of the traveling salesman problem”. *Operations Research* 12(5), 655–679.

Graham, R., Lawler, E., Lenstra, J., and Kan, A. R. (1979). “Optimization and approximation in deterministic sequencing and scheduling: A survey”. *Annals of Discrete Mathematics* 5, 287 –326.

Gurobi Optimization, L. (2018). *Gurobi Optimizer Reference Manual*.

Hamdi, I. and Loukil, T. (2017). “The permutation flowshop scheduling problem with exact time lags to minimise the total earliness and tardiness”. *International Journal of Operational Research* 28(1), 70–86.

Huo, Y., Li, H., and 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. and 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. and Casar-Corredera, J. R. (1994). “Optimal radar pulse scheduling using a neural network”. *IEEE International Conference on Neural Networks*. Vol. 7, 4588–4591.

Johnson, S. M. (1954). “Optimal two- and three-stage production schedules with setup times included”. *Naval Research Logistics Quarterly* 1(1), 61–68.

Lehoux-Lebacque, V., Brauner, N., and Finke, G. (2015). “Identical coupled task scheduling: polynomial complexity of the cyclic case”. *Journal of Scheduling* 18(6), 631–644.

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

Li, H. and Zhao, H. (2007). “Scheduling Coupled-Tasks on a Single Machine”. *IEEE Symposium on Computational Intelligence in Scheduling*, 137–142.

Liu, M., Liu, X., Zheng, F., and Chu, F. (2017). “Bi-objective optimization of a reentrant flow shop scheduling with exact time lag considering energy cost”. *7th International Conference on Industrial Engineering and Systems Management (IESM 2017)*. Saarbrücken, Germany.

McNaughton, R. (1959). “Scheduling with deadlines and loss functions”. *Management Science* 6(1), 1–12.

Meziani, N., Boudhar, M., and Oulamara, A. (2018a). “PSO and simulated annealing for the two-machine flowshop scheduling problem with coupled-operations”. *European Journal of Industrial Engineering* 12(1), 43–66.

Meziani, N., Oulamara, A., and Boudhar, M. (2018b). “Two-machine flowshop scheduling problem with coupled-operations”. *Annals of Operations Research*.

Mitten, L. G. (1959). “Sequencing n jobs on two machines with arbitrary time lags”. *Management Science* 5(3), 293–298.

Nawaz, M., Enscore, E. E., and Ham, I. (1983). “A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem”. *Omega* 11(1), 91 –95.

Orman, A. and Potts, C. (1997). “On the complexity of coupled-task scheduling”. *Discrete Applied Mathematics* 72(1), 141 –154.

Orman, A., Potts, C., Shahani, A., and Moore, A. (1996). “Scheduling for a multifunction phased array radar system”. *European Journal of Operational Research* 90(1), 13 –25.

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

Potts, C. N. and Van Wassenhove, L. N. (1982). “A decomposition algorithm for the single machine total tardiness problem”. *Operations Research Letters* 1, 177–181.

Potts, C. N. and Whitehead, J. D. (2007). “Heuristics for a coupled-operation scheduling problem”. *Journal of the Operational Research Society* 58(10), 1375–1388.

Röck, H. (1984). “Some new results in flow shop scheduling”. *Zeitschrift für Operations Research* 28(1), 1–16.

Sahni, S. and Cho, Y. (1979). “Complexity of scheduling shops with no wait in process”. *Mathematics of Operations Research* 4(4), 448–457.

Shapiro, R. D. (1980). “Scheduling coupled tasks”. *Naval Research Logistics Quarterly* 27(3), 489–498.

Sherali, H. D. and Smith, J. C. (2005). “Interleaving two-phased jobs on a single machine”. *Discrete Optimization* 2(4), 348 –361.

Simonin, G., Giroudeau, R., and König, J.-C. (2011a). “Complexity and approximation for scheduling problem for a torpedo”. *Computers & Industrial Engineering* 61(2), 352 –356.

Simonin, G., Darties, B., Giroudeau, R., and König, J.-C. (2011b). “Isomorphic coupled-task scheduling problem with compatibility constraints on a single processor”. *Journal of Scheduling* 14(5), 501–509.

Simonin, G. (2009). “Limpact de Iintroduction du graphe de compatibilit dans les problmes dordonnancement en prsence de tches-couples”. PhD thesis. Montpellier, France: Universite de Montpellier II.

Simonin, G., Giroudeau, R., and König, J.-C. (2013). “Approximating a coupled-task scheduling problem in the presence of compatibility graph and additional tasks”. *International Journal of Planning and Scheduling* 1(4), 285–300.

Yu, W., Hoogeveen, H., and 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(5), 333–348.

Zhang, X. and Van De Velde, S. (2010). “Polynomial-time approximation schemes for scheduling problems with time lags”. *Journal of Scheduling* 13(5), 553–559.