Computers & Industrial Engineering 183 (2023) 109456

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Flow-shop scheduling with exact delays to minimize makespan

Mostafa Khatami [a,b], Amir Salehipour [c,*], T.C.E. Cheng [d]

[a] School of Mathematical and Physical Sciences, University of Technology Sydney, NSW 2007, Australia
[b] Centre for Data Science, Queensland University of Technology, QLD 4000, Australia
[c] The University of Sydney Business School, The University of Sydney, Australia
[d] PolyU Business School, The Hong Kong Polytechnic University, Hong Kong

ARTICLE INFO

Keywords:
Scheduling
Flow-shop
Coupled task
Exact delays
Pyramidal property

ABSTRACT

The flow-shop scheduling problem with exact delays is generalization of no-wait flow-shop scheduling in which an exact delay exists between the consecutive tasks of each job. The problem with distinct delays to minimize the makespan is strongly NP-hard even for the two-machine case with unit execution time tasks. Providing polynomial-time solutions for special cases of the problem, we show that the two-machine permutation flow-shop case is solvable in $O(n \log n)$ time, while the case with more than two machines is strongly NP-hard. We also show that the multi-machine case with delays following the ordered structure possesses the pyramidal-shaped property and propose an $O(n^2)$-time dynamic program to solve it. We further improve the time complexity of the solution algorithm to $O(n \log n)$ under certain conditions.

1. Introduction

The flow-shop scheduling problem is one of the well-studied combinatorial optimization problems, where a given set J of jobs, each with m tasks, must be processed on m distinct machines. The flow-shop problem can be categorized into permutation and non-permutation variants, where the processing order of the jobs on all the m machines is the same in the permutation setting.

In addition to the classical flow-shop scheduling problem, there are numerous studies considering the problem in various settings under different processing conditions. A well-known variant is the "no-wait" flow-shop (Gilmore & Gomory, 1964), in which all tasks of a job must be processed without any delay, i.e., once the processing of a job starts, there is no waiting time between the processing of its tasks.

Khatami et al. (2020) discussed that no-wait shop scheduling is a special case of coupled task scheduling. In the coupled task setting, the tasks of a job must be processed with an "exact" time lag between them, meaning that there is an exact delay between the processing of any two consecutive tasks of a job. It is clear that the problem reduces the no-wait case if all the delays are equal to zero.

Another variant of the classical flow-shop scheduling problem is the "ordered" flow-shop introduced by Smith (1968), in which the processing time of the jobs follow two specific structures: (*i*) if the processing time of a job is smaller than that of another job on some machine, then it is the case on all the machines, and (*ii*) if the processing time of a job on a machine is smaller than its processing time on another machine, then it is the case for all the jobs. The "no-wait ordered" flow-shop scheduling problem includes both the ordered and no-wait features, i.e., the processing times of the tasks of the jobs follow the ordered conditions and there is no waiting time between the processing of the tasks of a job.

In this study we generalize the no-wait ordered flow-shop: We consider the coupled task flow-shop scheduling problem, which is also known as flow-shop scheduling with exact delays. We study the problem with the objective of minimizing the makespan, i.e., the maximum completion time of the jobs.

1.1. Review of computational complexity

In this section we focus on the main results obtained for the classical, no-wait, ordered, and coupled task variants of the flow-shop scheduling problem from the computational complexity lens. We discuss the results under the objective function of minimizing the makespan, which is also the focus of this paper.

The two-machine flow-shop scheduling problem is known to be polynomially solvable (Johnson, 1954). However, the problem with more than two machines is strongly NP-hard (Garey et al., 1976). Similarly, the no-wait flow-shop scheduling problem is polynomially solvable if there are only two machines (Gilmore & Gomory, 1964) and becomes strongly NP-hard for any number of machines greater than two (Röck, 1984). For reviews of permutation, non-permutation, and no-wait variants of the problem we refer the interested reader to

* Corresponding author.
E-mail addresses: mostafa.khatami@qut.edu.au (M. Khatami), amir.salehipour@sydney.edu.au (A. Salehipour), edwin.cheng@polyu.edu.hk (T.C.E. Cheng).

https://doi.org/10.1016/j.cie.2023.109456
Received 7 December 2020; Received in revised form 17 September 2022; Accepted 15 July 2023
Available online 22 July 2023

Table 1
Summary of major results on flow-shop scheduling with the no-wait, ordered, and coupled task features.

Problem	Coupled task	Ordered	Additional assumption	Complexity/result	Reference
$F2$	–	–	–	$O(n \log n)$	Johnson (1954)
	✓	–	Zero-delays (no-wait)	$O(n \log n)$	Gilmore and Gomory (1964)
	–	✓	–	$O(n \log n)$	Smith et al. (1975)
	✓	✓	Zero-delays (no-wait)	$O(n \log n)$	Panwalkar and Woollam (1979)
	✓	–	–	Strongly NP-hard	Yu et al. (2004)
	✓	–	Identical delays	$O(n \log n)$	Leung et al. (2007)
	✓	–	Permutation	$O(n \log n)$	**This paper**
F	–	–	–	Strongly NP-hard	Garey et al. (1976)
	✓	–	Zero-delays (no-wait)	Strongly NP-hard	Röck (1984)
	–	✓	–	$O(n \log n)$[a]	Smith et al. (1975)
	–	✓	–	Pyramidal property	Smith et al. (1976)
	✓	✓	Zero-delays (no-wait)	$O(n \log n)$[a]	Panwalkar and Woollam (1979)
	✓	✓	Zero-delays (no-wait)	Pyramidal property	Arora and Rana (1980)
	✓	–	Permutation	Strongly NP-hard	**This paper**
	✓	✓	Ordered delays, permutation	Pyramidal property, $O(n^2)$	**This paper**
	✓	✓	Ordered delays, permutation	$O(n \log n)$[a]	**This paper**

[a]If the largest processing times occur on the first or last machine.

Allahverdi (2016), Hejazi and Saghafian (2005), and Rossit et al. (2018) respectively.

The coupled task flow-shop problem to minimize the makespan has been mostly studied for the two-machine case. Leung et al. (2007) showed that the two-machine case can be solved in $O(n \log n)$ time if all the delays are equal. However, the problem is intractable if arbitrary delays are considered. For example, Yu et al. (2004) showed that even if only two distinct values are considered for the delays, then the problem is strongly NP-hard. This result holds even for the case with unit execution time (UET) tasks. The interested reader is referred to Khatami et al. (2020) for a comprehensive review of the coupled task scheduling problem.

In the ordered flow-shop scheduling problem where the longest processing times occur on the first machine, Smith et al. (1975) showed that the longest processing time (LPT) first dispatching rule is optimal. Similarly, the shortest processing time (SPT) first dispatching rule is optimal if the longest processing times occur on the last machine. Smith et al. (1976) showed that a pyramidal-shaped sequence is optimal for the ordered flow-shop scheduling problem. A pyramidal-shaped sequence consists of two sets of jobs. The jobs in the first set are sequenced in the SPT order and the jobs in the second set are sequenced in the LPT order. In addition, for makespan minimization, the LPT order is optimal if the largest task of every job occurs on the first machine and the SPT order is optimal if the last machine processes the largest tasks of all the jobs. We refer the interested reader to Khatami et al. (2019) for the latest developments of research on the ordered flow-shop scheduling problem.

For the no-wait ordered problem, Panwalkar and Woollam (1979) showed that sequencing the jobs in the LPT (SPT) order minimizes the makespan if the largest processing times occur on the first (last) machine. Later, Arora and Rana (1980) established the pyramidal property of the problem. It is evident that the coupled task ordered flow-shop scheduling problem is generalization of the no-wait ordered flow-shop scheduling problem.

The contributions of this paper is twofold. First, the results distinguish between easy and hard cases of the problem: The problem is easy with two machines, and hard with more than two machines. Therefore, our results set the borderline between the NP-hard and polynomial cases, with respect to the number of machines. Second, we derive new properties and polynomial-time algorithms for the case of ordered delays with arbitrary number of machines.

Table 1 summarizes the major results on flow-shop scheduling with the no-wait, ordered, and coupled task features, as well as our contributions. In Table 1 $F2$ and F denote flow-shop scheduling with two and an arbitrary number of machines, respectively.

1.2. Applications

The importance of the coupled task scheduling problem is twofold. First, from the theoretical point of view, the coupled task scheduling problem is generalization of the no-wait flow-shop scheduling problem. Second, the coupled task scheduling problem has various real-world applications in different sectors. The classical applications include modeling of a pulsed radar system (Shapiro, 1980), the scheduling problem to process the environmental data in submarine torpedoes (Simonin et al., 2011), modeling of certain chemical manufacturing processes (Ageev & Baburin, 2007), and the scheduling problem in a single-machine no-wait robotic cell system (Brauner et al., 2009; Lehoux-Lebacque et al., 2015).

Recently, the coupled task scheduling problem has been utilized to model certain problems in the healthcare domain. For example, to schedule chemotherapy appointments, once the medicine is prescribed the patient visits the health centre on treatment days separated by a fixed number of rest days (Condotta & Shakhlevich, 2014). Some healthcare environments with multi-stage characteristics can also be modeled as the coupled task scheduling problem. One example includes the radiation therapy clinics where patients require a number of sequential treatments with fixed delays between them (Saure et al., 2012). Consider a pathology laboratory, where minimizing the patients' waiting times is the performance measure (Azadeh et al., 2014; Marinagi et al., 2000). Certain blood tests, e.g., fasting blood sugar testing, require multiple tests and there is an exact time delay between a pair of tests. Another example includes patient appointment scheduling in nuclear medicine clinics. Due to the very strict multi-stage sequential procedures, the problem in the nuclear medicine clinic is more complex than its counterpart in the typical medical imaging clinic (Pérez et al., 2013, 2011). Here, a single treatment requires multiple steps and each step needs to be completed within a strict time window. Maximizing the number of treated patients is well justified in this context due to the costs of the required resources and the short half-lives of the radio-pharmaceuticals needed for the treatments. Those healthcare applications become complex when staff tiredness that typically occurs in practice and impacts the treatment times are considered. Khatami and Salehipour (2021) proposed time-dependent treatment times to address human fatigue within the coupled task scheduling problem.

The ordered setting is also applicable in the health care environment. For example, in radiation therapy treatments, a specific regimen includes a number of treatments with equal lengths (Saure et al., 2012), i.e., the jobs are ordered as in condition (*i*) discussed above. In the same problem, consultations take longer than treatment sessions (Bikker et al., 2015), i.e., the machines are ordered as in condition (*ii*) discussed above.

Table 2
Data for problem $I_{2\times2}$.

Job	p_{1j}	p_{2j}	L_{1j}
1	1	2	3
2	2	3	6

1.3. Paper contributions and organization

Our contributions include (1) presenting the coupled task flow-shop problem with an arbitrary number of machines as the traveling salesman problem, (2) establishing that the two-machine permutation flow-shop case is equivalent to the no-wait case and showing that it is polynomially solvable, (3) investigating the case with both distinct and ordered delays, in addition to the case with ordered processing times, (4) developing several properties of the latter case and showing that the optimal schedule possesses the pyramidal-shaped property, and proposing an $O(n^2)$-time dynamic programming solution algorithm, and (5) further improving the time complexity of the solution algorithm to $O(n \log n)$ for two specific cases.

We organize the rest of the paper as follows: In Section 2 we formally define the problem and formulate it as the traveling salesman problem. We investigate the case with distinct delays in Section 3 and find the optimal makespan for the two-machine case. In addition, we show that the problem becomes strongly NP-hard when there are more than two machines. We devote Section 4 to studying the case with ordered delays, showing that the optimal permutation schedule possesses the pyramidal-shaped property, and providing an $O(n^2)$ dynamic program to solve the general problem and an $O(n \log n)$ procedure to solve the case where the largest processing times occur on the first or the last machine. Finally, in Section 5, we conclude the paper and suggest topics for future research.

2. Problem definition and representation

The coupled task flow-shop scheduling problem concerns scheduling a set $J = \{1, \dots, n\}$ of jobs on a set $M = \{1, \dots, m\}$ of distinct machines. Each job j consists of m tasks to be processed on the m machines, where all the jobs follow the same machine route $1, 2, \dots, m$. There is an exact delay between every two consecutive tasks of the same job. Therefore, a succeeding task is started after the completion of its preceding task plus an exact amount of delay. The processing time of job $j \in J$ on machine $r \in M$ is a known positive integer and is denoted by $p_{rj} \in \mathbf{Z}^+$. Also, the duration of the delay of job j after being processed on machine $r \in M \setminus \{m\}$ is denoted by $L_{rj} \in \mathbf{Z}^+$. We assume that all the jobs are available at time zero and preemption of the tasks is not allowed, i.e., once the processing of a task is started, it cannot be interrupted by other tasks. Each machine can process at most one task at a time. The objective is to minimize the makespan, i.e., minimizing the completion time of the last job in the sequence of executing the jobs on the machines.

We now discuss several properties of the coupled task flow-shop scheduling problem. First, we note that the coupled task flow-shop reduces to the no-wait flow-shop if $L_{rj} = 0, \forall j \in J, r \in M \setminus \{m\}$. Second, it is easy to see that only permutation schedules, in which the processing orders of the jobs on all the m machines are the same, are feasible for the no-wait flow-shop scheduling problem. However, in the coupled task flow-shop, non-permutation schedules can also be feasible. Therefore, the optimal schedule for the coupled task flow-shop scheduling problem may not necessarily be a permutation schedule.

Consider the instance with two machines and two jobs that we denote as problem $I_{2\times2}$. Table 2 shows the data for problem $I_{2\times2}$. We show in Fig. 1 the optimal makespan for problem $I_{2\times2}$, under both permutation and non-permutation schedules. It is evident that the non-permutation schedule (Fig. 1b) yields a makespan of 11, which is better than that under the permutation schedule, i.e., 12. We have the following lemma:

Table 3
The distance matrix of the TSP for the coupled task flow-shop scheduling problem.

–	D_{12}	D_{13}	…	D_{1n}	T_1
D_{21}	–	D_{23}	…	D_{2n}	T_2
.	.				.
.		.			.
.			.		.
D_{n1}	D_{n2}	D_{n3}	…	–	T_n
0	0	0	…	0	–

Lemma 1. *The optimal schedule for the coupled task flow-shop scheduling problem is not necessarily a permutation schedule.*

Proof. See the optimal schedule for problem $I_{2\times2}$ depicted in Fig. 1. □

In the rest of the paper we only consider permutation schedules for the coupled task flow-shop problem because the problem with non-permutation schedules is strongly NP-hard even for the case with two machines, two distinct delays, and UET tasks (Yu et al., 2004). Next, we formulate the coupled task flow-shop scheduling problem as the traveling salesman problem (TSP), and develop some properties of the problem based on the TSP formulation in Sections 3 and 4.

Baker and Trietsch (2013) showed that the n-job no-wait flow-shop scheduling problem can be formulated as an $(n+1)$-city TSP, where each city corresponds to a job and the intercity distances correspond to the delays between the jobs. A dummy city, from which the distances to all the other cities are zero is added. The distance from city j to the dummy city is the sum of the processing times of job j. We formulate the coupled task flow-shop scheduling problem as TSP because the problem is generalization of the no-wait flow-shop scheduling problem.

Let D_{jk} be the delay in starting job k, measured from the start time of job j, which is calculated by

$$D_{jk} = p_{1j} + \max\Big\{0, (p_{2j} - p_{1k} + L_{1j} - L_{1k}), (p_{2j} + p_{3j} - p_{1k} - p_{2k} + L_{1j} + L_{2j} - L_{1k} - L_{2k}), \dots, \Big(\sum_{r=2}^{m} p_{rj} - \sum_{r=1}^{m-1} p_{rk} + \sum_{r=1}^{m-1} L_{rj} - \sum_{r=1}^{m-1} L_{rk}\Big)\Big\}, \quad \forall j, k \in J. \tag{1}$$

In Eq. (1), the delay in starting job k after job j in the sequence is composed of two terms. The first term is the processing time of job j on the first machine since the delay is calculated from the start time of job j. The second term calculates the maximum delay incurred on the start time of job j on machines 2 to m. In addition, let T_j be the distance from city j to the dummy city, i.e.,

$$T_j = \sum_{r=1}^{m} p_{rj} + \sum_{r=1}^{m-1} L_{rj}. \tag{2}$$

Then, we show in Table 3 the distance matrix of the corresponding TSP. From the distance matrix shown in Table 3, we see that a tour for the TSP corresponds to a sequence of processing the jobs on the machines. The total cost of the tour is equivalent to the makespan of the sequence.

We can use a "reduced" distance matrix to simplify the TSP formulation of the coupled task flow-shop scheduling problem. To this end, we define D'_{jk} and T'_j via Eqs. (3) and (4) as follows:

$$D'_{jk} = D_{jk} - p_{1j}, \forall j, k \in J, \tag{3}$$

$$T'_j = T_j - p_{1j}, \forall j \in J. \tag{4}$$

Replacing D_{jk} with $D'_{jk}, \forall j, k \in J$, and T_j with $T'_j, \forall j \in J$ in Table 3, we obtain the reduced distance matrix. It is noted that the TSP tour remains the same under the reduced distance matrix. The optimal cost (makespan), however, is different by a constant value, which is equal to $\sum_{j=1}^{n} p_{1j}$. Next, we present our results for the coupled task flow-shop problem with distinct delays.

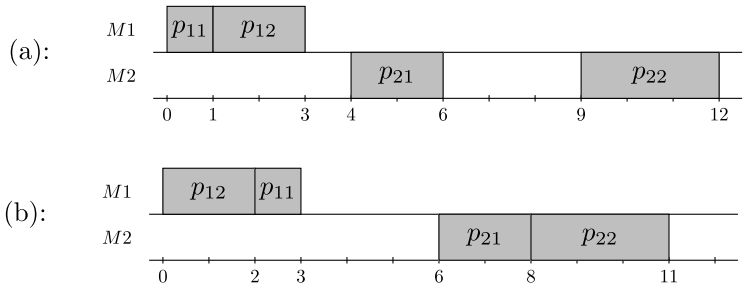

Fig. 1. The optimal permutation (a) and non-permutation (b) schedules for problem $I_{2\times2}$.

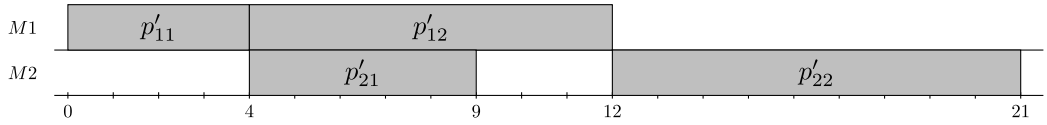

Fig. 2. The equivalent no-wait schedule to the optimal schedule for case P.

3. Distinct delays

Recall that the coupled task flow-shop scheduling problem with non-permutation schedules is strongly NP-hard even for the two-machine case. However, the two-machine case with permutation schedules and identical delays is polynomially solvable (Gilmore & Gomory, 1964; Leung et al., 2007). We generalize this result to the case with distinct delays and show that it is polynomially solvable.

We first transform the case with distinct delays to the equivalent no-wait case, i.e., with zero delays. We let P and P′ denote the cases with distinct and zero delays, respectively. We set the processing time of each task in P′ as the processing time of the task in P plus the amount of delay of the task. The incurred delays between the jobs, i.e., D_{jk}, are equal in both cases. First, observe that minimizing the makespan for P′ is equivalent to minimizing the makespan for P. Second, it is clear that minimizing the makespan is equivalent to minimizing the total idle time on either machine or, equivalently, minimizing the total idle time on both machines (Emmons & Vairaktarakis, 2012). We let I_1 and I_2 denote the total idle times on machines 1 and 2, respectively. Third, note that irrespective of the processing times of P′, the values of idle times in both cases of P and P′ are equal.

Consider again problem $I_{2\times2}$. In the optimal schedule depicted in Fig. 1a, $I_1 = 9$ and $I_2 = 4+3 = 7$. It follows that $p'_{11} = p_{11} + L_{11} = 1+3 = 4, p'_{21} = p_{21} + L_{11} = 2+3 = 5, p'_{12} = p_{12} + L_{12} = 2+6 = 8$, and $p'_{22} = p_{22} + L_{12} = 3+6 = 9$, where p'_{rj} is the processing time of job j on machine r in case P′. We show in Fig. 2 the schedule for P′, which is equivalent to the optimal schedule for case P.

Gilmore and Gomory (1964) provided an $O(n\log n)$-time algorithm that finds the optimal makespan for case P′. Therefore, it suffices to transform case P to case P′, as stated in the following lemma.

Lemma 2. *The two-machine coupled task scheduling problem with distinct delays and permutation schedules can be transformed into an equivalent two-machine no-wait flow-shop scheduling problem.*

Proof. Let $p'_{rj} = p_{rj} + L_{1j}, \forall j \in J, r = 1,2$. Substituting them into Eq. (1) yields $D_{jk} = p_{1j} + \max\{0, (p'_{2j} - p'_{1k})\}, \forall j,k \in J$, which is indeed the definition of the no-wait flow-shop problem given in Baker and Trietsch (2013). As a result, the case with distinct delays and processing times p_{rj} is equivalent to the no-wait case with processing times p'_{rj}. □

Lemma 2 results in Theorem 1.

Theorem 1. *The optimal makespan for the two-machine coupled task flow-shop scheduling problem with distinct delays and permutation schedules can be obtained in $O(n\log n)$ time.*

Proof. The result is clear because the transformation is performed in constant time (see Lemma 2) and the two-machine no-wait flow-shop problem can be solved in $O(n\log n)$ time by the algorithm of Gilmore and Gomory (1964). □

We observe that Theorem 1 does not hold when there are more than two machines (see the definition in Eq. (1)). Indeed, the problem is not trivial for $m > 2$:

Theorem 2. *The coupled task flow-shop scheduling problem with distinct delays and permutation schedules is strongly NP-hard for $m > 2$.*

Proof. We immediately deduce the result because the coupled task flow-shop problem is a generalization of the no-wait flow-shop problem, which is strongly NP-hard for $m > 2$ as shown in Röck (1984). □

4. Ordered delays

In this section we study the coupled task flow-shop problem with the additional assumption of ordered processing times and delays. The following two conditions are satisfied for the ordered processing times assumption:

(*i*) for any two jobs $j,k \in J$, if $p_{rj} < p_{rk}, r \in M$, then $p_{qj} \le p_{qk}, \forall q \in M$; and,
(*ii*) for any two machines $r,q \in M$, if $p_{rk} < p_{qk}, k \in J$, then $p_{rj} \le p_{qj}, \forall j \in J$,

where (*i*) is called the job-ordered condition and (*ii*) is called the machine-ordered condition. Following the job-ordered condition, we can rank the jobs by their processing times as follows: We call job j smaller than job k, which we denote by $j < k$, if the processing time of job k on each machine is at least as large as that of job j on the same machine. For the case of two identical jobs, we rank them by their job indices.

We can show that the problem with ordered processing times and distinct delays is strongly NP-hard even for the case with two machines. The proof follows from the case with UET tasks, which satisfies the assumption of ordered processing times and is strongly NP-hard (Yu et al., 2004). If delays are identical, however, the problem is polynomially solvable (Gilmore & Gomory, 1964; Leung et al., 2007). So we consider the case with ordered delays: If job j is smaller than job k, i.e., $j < k$, then the delay of job k after completion on any machine $r < m$ is at least as large as the delay of job j after completion on the same machine. We formally state this condition as follows:

(*iii*) for any two jobs $j,k \in J$, if $j < k$, then $L_{rj} \le L_{rk}, \forall r \in M \setminus \{m\}$.

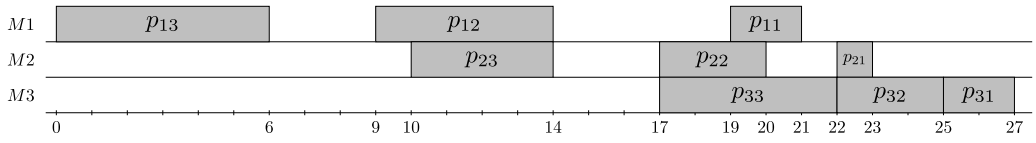

Fig. 3. The optimal schedule for problem $I_{3\times3}$.

Table 4
The data for problem $I_{3\times3}$.

Job	p_{1j}	p_{2j}	p_{3j}	L_{1j}	L_{2j}
1	2	1	2	1	2
2	5	3	3	3	2
3	6	4	5	4	3

Table 5
The reduced distance matrix of TSP for problem $I_{3\times3}$.

–	0	0	**6**
5	–	0	11
10	**3**	–	16
0	0	**0**	–

Therefore, in addition to conditions (*i*) and (*ii*), the new condition (*iii*) ensures that the delays are ordered as well. We call this case coupled task ordered flow-shop scheduling. Observe from (*iii*) that the case with identical delays is a special case of ordered delays.

We present the reduced distance matrix of the TSP associated with the coupled task ordered flow-shop scheduling problem via an example, which we denote by problem $I_{3\times3}$. Problem $I_{3\times3}$ consists of three jobs and three machines labeled as M1, M2, and M3. Table 4 shows the data of problem $I_{3\times3}$.

As shown in Table 4, the jobs are ordered as $1 < 2 < 3$ and the machines are ordered as $M1 > M3 > M2$. The delays are also ordered. Table 5 shows the reduced distance matrix of the TSP associated with problem $I_{3\times3}$, where the values appearing in the optimal tour are highlighted. The minimum makespan is equal to 27 and the optimal job sequence is $(3, 2, 1)$, which are depicted in Fig. 3 and are obtained by solving the TSP. We note that an instance of the TSP needs to be solved to find the optimal sequence.

Next, we derive an important property of an optimal sequence for the coupled task ordered flow-shop scheduling problem, and then we find an optimal schedule for the problem.

4.1. The pyramidal property

We show that the pyramidal-shaped property holds for an optimal sequence for the coupled task ordered flow-shop scheduling problem. The pyramidal-shaped property implies that the jobs can be partitioned into two disjoint sets, where the jobs in the first set are sequenced in the SPT order and those in the second set follow the LPT order. For example, based on problem $I_{3\times3}$, sequence (1, 3, 2) is a pyramidal sequence as the processing time of job 3 is longer than those of jobs 1 and 2 on any machine, i.e., job 3 is larger than jobs 1 and 2. On the other hand, sequence (3, 1, 2) is not a pyramidal sequence. Without loss of generality, we assume that the largest job is in the first set. We start by showing the following property.

Property 1. *For any four jobs $j, k \in J, j < k$ and $i, l \in J$, and $i < l$, $D'_{ki} - D'_{kl} \geq D'_{ji} - D'_{jl}$.*

Proof. We represent the elements of the reduced distance matrix as follows:

$$D'_{jk} = \max\{0, \zeta_1, \dots, \zeta_{m-1}\},$$

where $\zeta_1 = (p_{2j} - p_{1k} + L_{1j} - L_{1k})$, as defined in Eq. (1), and so on for the rest of the elements. By substitution, we have

$$\begin{aligned}&\max\{0, A_1, A_2, \dots, A_{m-1}\} - \max\{0, B_1, B_2, \dots, B_{m-1}\} \geq \\ &\quad \max\{0, C_1, C_2, \dots, C_{m-1}\} - \max\{0, E_1, E_2, \dots, E_{m-1}\},\end{aligned} \tag{5}$$

where A, B, C, and E represent the elements of $D'_{ki}, D'_{kl}, D'_{ji}$, and D'_{jl}, respectively. We call two elements "similar" if they have the same index, e.g., A_2 and B_2, and "non-similar" otherwise. It is evident that if all the four maximum terms are equal to zero, the property holds. The property also holds if in D'_{ki} (or in D'_{jl}), the maximum term is larger than zero, while the other three maximum terms are equal to zero. Therefore, we consider the case where in D'_{kl} the maximum term is larger than zero, while the other three maximum terms are equal to zero.

For that, suppose the second element is the largest, i.e., $D'_{kl} = B_1$. The assumption results in $0 - B_1 \geq 0 - 0$, meaning that the property holds if and only if $B_1 \leq 0$. Suppose $B_1 > 0$, which leads to $p_{2k} - p_{1l} + L_{1k} - L_{1l} > 0$. Because $i < l$, we have $p_{1i} < p_{1l}$ and $L_{1i} < L_{1l}$. Therefore,

$$\begin{aligned}&p_{2k} - p_{1l} + L_{1k} - L_{1l} > 0, \implies p_{2k} + L_{1k} > p_{1l} + L_{1l}, \\ &\implies p_{2k} + L_{1k} > p_{1i} + L_{1i}, \implies p_{2k} - p_{1i} + L_{1k} - L_{1i} > 0.\end{aligned} \tag{6}$$

Observe that $p_{2k} - p_{1i} + L_{1k} - L_{1i}$ is the second element, i.e., A_1 in the first maximum term. Based on our earlier assumption that the first maximum term is equal to zero, none of its elements including A_1 can be positive, so a contradiction. Similar calculations can be performed for the other elements of D'_{kl} and D'_{ji}. Similarly, for the cases where two or three maximum terms are equal to zero, the property can be shown to hold.

Now, let us consider the case where all of the maximum terms in inequality (5) are larger than zero. We start with the case where similar elements are the largest in all the four maximum terms. For example, consider the case where the second elements are the largest in all the maximum terms. So we must show that $A_1 - B_1 \geq C_1 - E_1$. By expanding the terms, we obtain

$$\begin{aligned}&(p_{2k} - p_{1i} + L_{1k} - L_{1i}) - (p_{2k} - p_{1l} + L_{1k} - L_{1l}) \geq \\ &\quad (p_{2j} - p_{1i} + L_{1j} - L_{1i}) - (p_{2j} - p_{1l} + L_{1j} - L_{1l}),\end{aligned} \tag{7}$$

where all the elements are cancelled out on both sides and the property always holds. This can also be shown for any other element. Now we consider the cases where the non-similar elements are the largest. We start with the case where the largest elements in exactly three of the terms are similar. For example, assume that B_1, C_1, and E_1 are the largest elements in their respective terms; however, A_2 is the largest element of D'_{ki}, implying that $A_2 \geq A_1$, so

$$\begin{aligned}&(p_{2k} + p_{3k} - p_{1i} - p_{2i} + L_{1k} + L_{2k} - L_{1i} - L_{2i}) \geq (p_{2k} - p_{1i} + L_{1k} - L_{1i}) \\ &\implies (p_{3k} - p_{2i} + L_{2k} - L_{2i}) \geq 0.\end{aligned} \tag{8}$$

It suffices to show that $A_2 - B_1 \geq C_1 - E_1$. Expanding the terms, we obtain

$$\begin{aligned}&(p_{2k} + p_{3k} - p_{1i} - p_{2i} + L_{1k} + L_{2k} - L_{1i} - L_{2i}) - (p_{2k} - p_{1l} + L_{1k} - L_{1l}) \geq \\ &\quad (p_{2j} - p_{1i} + L_{1j} - L_{1i}) - (p_{2j} - p_{1l} + L_{1j} - L_{1l}) \\ &\implies (p_{3k} - p_{2i} + L_{2k} - L_{2i}) \geq 0,\end{aligned} \tag{9}$$

which holds because it follows from our assumption in inequality (8). Considering any other largest term of A will lead to the same result. Similar calculations can be performed for D'_{jl}.

We also show that the property holds if A_1, C_1, and E_1 are the largest elements in their respective terms, and B_2 is the largest element in D'_{kl}. The latter leads to $(p_{3k} - p_{2l} + L_{2k} - L_{2l}) \geq 0$ following inequality (8). Expanding the terms, we obtain

$$\begin{aligned}&(p_{2k} - p_{1i} + L_{1k} - L_{1i}) - (p_{2k} + p_{3k} - p_{1l} - p_{2l} + L_{1k} + L_{2k} - L_{1l} - L_{2l}) \geq \\ &\quad (p_{2j} - p_{1i} + L_{1j} - L_{1i}) - (p_{2j} - p_{1l} + L_{1j} - L_{1l}) \\ &\quad \Longrightarrow \; -(p_{3k} - p_{2i} + L_{2k} - L_{2i}) \geq 0,\end{aligned} \tag{10}$$

which implies that either $(p_{3k} - p_{2i} + L_{2k} - L_{2i}) = 0$, where the property holds, or $(p_{3k} - p_{2i} + L_{2k} - L_{2i}) < 0$, which is a contradiction to our assumption. Considering any other largest term of B will lead to the same result, and we can apply a similar argument for D'_{ji}. This completes the proof.

Noting that considering any other combination of the largest elements in the terms is similar to one of the aforementioned cases, we omit the proof for brevity. □

Property 1 leads to the following theorem.

Theorem 3. *An optimal sequence for the coupled task ordered flow-shop scheduling problem is in the pyramidal shape.*

Proof. We claim that any non-pyramidal sequence is dominated by a pyramidal one. Let π denote a pyramidal sequence for a set of n jobs, where job n is the largest job, i.e., it has the longest processing time. It is clear that $\pi = (a, \dots, l, n, k, \dots, b)$, where $a < \cdots < l < n$ and $n > k > \cdots > b$. Let $\pi = \pi_L \cup \pi_R$, where $\pi_L = (a, \dots, l, n)$ and $\pi_R = (k, \dots, b)$, i.e., we assume that job n is in π_L. Any shuffling of the job order of either π_L or π_R leads π to be a non-pyramidal sequence. Let π' present such a non-pyramidal sequence and π'_L correspond to the first part of the sequence, i.e., the sequence of the jobs up to and including job n and π'_R denotes the second part of the sequence, i.e., the sequence of the jobs after job n. We need to show that π is derivable from π' and that such a process does not increase the makespan, so the makespan of π' is never better than that of π.

In order to derive π from π', we follow the rule of Arora and Rana (1980). We select the next largest job in $\pi'_L \setminus \{n\}$ and move it to a position before a job just larger than it in the same sequence π'_L. We show that the move does not increase the makespan. Assume that, at some stage, the tour associated with π'_L is

$$S'_L = (n+1, 1, 2, \dots, a, j, b, \dots, c, j+1, \dots, l, n),$$

where $n+1$ and n are the dummy city and the largest job, respectively. Also, j is the next largest job in $\pi'_L \setminus \{n\}$ and $j+1$ is the job larger than j. We denote the total distance for tour S'_L as z'_L:

$$z'_L = 0 + D'_{12} + \cdots + D'_{aj} + D'_{jb} + \cdots + D'_{c,j+1} + \cdots + D'_{ln}.$$

According to the rule, job j is moved to the position between jobs c and $j+1$. Let $z''_L = 0 + D'_{12} + \cdots + D'_{ab} + \cdots + D'_{cj} + D'_{j,j+1} + \cdots + D'_{ln}$ denote the makespan obtained as the result of the move. We need to show that $z'_L - z''_L = (D'_{aj} + D'_{jb} + D'_{c,j+1}) - (D'_{ab} + D'_{cj} + D'_{j,j+1}) \geq 0$. Either of the following two cases is possible: (1) $c > a$ or (2) $c < a$. We now show that in both cases, the makespan either improves or remains the same.

Case 1: Assume that $c > a$. Adding and subtracting D'_{cb} to and from the inequality $z'_L - z''_L = (D'_{aj} + D'_{jb} + D'_{c,j+1} + D'_{cb}) - (D'_{ab} + D'_{cj} + D'_{j,j+1} + D'_{cb}) \geq 0$ and re-arranging the terms yields

$$(D'_{cb} - D'_{cj}) + (D'_{aj} - D'_{ab}) + (D'_{jb} - D'_{j,j+1}) + (D'_{c,j+1} - D'_{cb}) \geq 0.$$

Given that $j+1 > j, j > a, b, c$, and $c > a$ (as the underlying assumption of case 1), we have

$$D'_{cb} - D'_{cj} \geq D'_{ab} - D'_{aj}$$

and

$$D'_{jb} - D'_{j,j+1} \geq D'_{cb} - D'_{c,j+1},$$

which hold due to Property 1. Hence, $z'_L - z''_L \geq 0$.

Case 2: Assume that $c < a$. Again after adding and subtracting $D'_{a,j+1}$ to and from the inequality and re-arranging the terms, we have $(D'_{jb} - D'_{j,j+1}) + (D'_{a,j+1} - D'_{ab}) + (D'_{aj} - D'_{a,j+1}) + (D'_{c,j+1} - D'_{cj})$. It follows that

$$D'_{jb} - D'_{j,j+1} \geq D'_{ab} - D'_{a,j+1}$$

and

$$D'_{aj} - D'_{a,j+1} \geq D'_{cj} - D'_{c,j+1},$$

which hold due to Property 1, leading to $z'_L - z''_L \geq 0$. We note that similar calculations can be performed for π'_R, i.e., to the LPT part of the non-pyramidal sequence π'. As a result, any non-pyramidal sequence π' is dominated by the pyramidal sequence π. This completes the proof. □

Theorem 3 is the basis for a polynomial-time algorithm to find the minimum makespan for the coupled task ordered flow-shop scheduling that we propose in the next section.

4.2. The optimal makespan

We can find an optimal sequence for the coupled task ordered flow-shop scheduling problem in $O(n^2)$ time. In addition, we show that, under certain conditions, we can find an optimal sequence in $O(n \log n)$ time.

Theorem 4. *The minimum makespan for the coupled task ordered flow-shop scheduling problem can be found in $O(n^2)$ time.*

Proof. We prove the theorem by noting that the problem can be formulated as a TSP and that finding the shortest pyramidal tour in the TSP can be performed in $O(n^2)$ by dynamic programming (Burkard et al., 1998; van der Veen & van Dal, 1991). The TSP tour $S = (n+1, a, \dots, l, n, k, \dots, b, n+1)$ associated with π is called a pyramidal tour if the distances between cities a and l are in ascending order, i.e., $a < \cdots < l$, and those between k and b are in descending order, i.e., $k > \cdots > b$. □

Next, we prove that if the first or the last machine processes the largest job, which we denote by M_1^{max} and M_m^{max}, respectively, then an optimal sequence can be efficiently found in $O(n \log n)$ time.

For the proof, we first present a few properties of the problem. We sort the jobs in non-decreasing order of their processing times and re-label them accordingly.

Property 2. $D'_{kj} \geq D'_{k,k-1} + D'_{k-1,k-2} + \cdots + D'_{j+1,j}, \forall j, k \in J, j < k, M_1^{max}$.

Proof. First, we consider $j = k-2$, which leads to $D'_{k,k-2} \geq D'_{k,k-1} + D'_{k-1,k-2}$. From Eqs. (1) and (3), we have

$$\begin{aligned}&(p_{2k} - p_{1,k-2} + L_{1k} - L_{1,k-2}) \geq (p_{2k} - p_{1,k-1} \\ &\quad + L_{1k} - L_{1,k-1}) + (p_{2,k-1} - p_{1,k-2} + L_{1,k-1} - L_{1,k-2}) \\ &\quad \Longrightarrow \; p_{2,k-1} - p_{1,k-1} \leq 0,\end{aligned}$$

which is always true. It is not difficult to show that the property holds for other values of j. □

Property 3. $T'_2 - T'_1 \geq D'_{21}, M_1^{max}$.

Proof. Using Eqs. (2) and (4) and expanding T'_2 and T'_1, we obtain

$$T'_2 - T'_1 = \Big(\sum_{r=1}^{m} p_{r2} + \sum_{r=1}^{m-1} L_{r2} - p_{12}\Big) - \Big(\sum_{r=1}^{m} p_{r1} + \sum_{r=1}^{m-1} L_{r1} - p_{11}\Big) =$$

$$\Big(\sum_{r=2}^{m} p_{r2} - \sum_{r=2}^{m} p_{r1} + \sum_{r=1}^{m-1} L_{r2} - \sum_{r=1}^{m-1} L_{r1}\Big),$$

which is always non-negative since $\sum_{r=2}^{m} p_{r2} \geq \sum_{r=2}^{m} p_{r1}$ and $\sum_{r=1}^{m-1} L_{r2} \geq \sum_{r=1}^{m-1} L_{r1}$.

It is also clear that the right-hand side is non-negative and can be expanded as

$$\max\{0, (p_{22} - p_{11} + L_{12} - L_{11}), (p_{22} + p_{32} - p_{11} - p_{21} + L_{12} + L_{22} - L_{11} - L_{21}), \dots, (\sum_{r=2}^{m} p_{r2} - \sum_{r=1}^{m-1} p_{r1} + \sum_{r=1}^{m-1} L_{r2} - \sum_{r=1}^{m-1} L_{r1})\}.$$

We need to show that the left-hand side is greater than or equal to every element inside the maximum term. We start with the term $(p_{22} - p_{11} + L_{12} - L_{11})$. As the jobs are sorted in non-decreasing order of their processing times, the left-hand side is at least as large as that term:

$$(\sum_{r=2}^{m} p_{r2} - \sum_{r=2}^{m} p_{r1} + \sum_{r=1}^{m-1} L_{r2} - \sum_{r=1}^{m-1} L_{r1}) \geq (p_{22} - p_{11} + L_{12} - L_{11}) \implies (\sum_{r=3}^{m} p_{r2} - \sum_{r=3}^{m} p_{r1}) + (\sum_{r=2}^{m-1} L_{r2} - \sum_{r=2}^{m-1} L_{r1}) \geq 0,$$

which is always true.

We can also show that the left-hand side is greater than or equal to the next element as well, i.e.,

$$(\sum_{r=2}^{m} p_{r2} - \sum_{r=2}^{m} p_{r1} + \sum_{r=1}^{m-1} L_{r2} - \sum_{r=1}^{m-1} L_{r1}) \geq (p_{22} + p_{32} - p_{11} - p_{21} + L_{12} + L_{22} - L_{11} - L_{21}).$$

This can be written as $(\sum_{r=4}^{m} p_{r2} - \sum_{r=4}^{m} p_{r1}) + (\sum_{r=3}^{m-1} L_{r2} - \sum_{r=3}^{m-1} L_{r1}) \geq 0$, which is also always true. Similarly, we can show that the left-hand side is at least as large as the other terms in the maximum term of the right-hand side. This completes the proof. □

We note that Property 3 can be further generalized to $T'_j - T'_1 \geq D'_{j1}, 2 \leq j \leq n, M_1^{max}$, but we do not give the proof here for brevity. Observe that Properties 2 and 3 hold for problem $I_{3\times3}$ and its reduced distance matrix, which is shown in Table 5. Properties 2 and 3 result in the following theorem.

Theorem 5. *The minimum makespan for the coupled task ordered flow-shop scheduling problem is obtained by the LPT sequence if M_1^{max}.*

Proof. Due to the pyramidal-shaped property of the problem discussed in Theorem 3, it suffices to show that the largest job is the first job in an optimal sequence. Suppose the tour associated with the LPT sequence is

$$S = (n+1, n, n-1, \dots, j, \dots, 2, 1, n+1),$$

where the total distance for the tour S, denoted as z_S, is equal to

$$z_S = 0 + D'_{n,n-1} + \cdots + D'_{j+1,j} + D'_{j,j-1} + \cdots + D'_{21} + T'_1.$$

To prove that the LPT sequence is optimal, we show that any change in tour S will not improve the makespan. We first consider the case of constructing a new tour S' by inserting a job $j, \forall j \in J \setminus \{n\}$, before the largest job n. We need to show that the total distance of tour S' is at least as large as that of tour S. We consider the following two cases.

Case 1. $j > 1$: $z_{S'}$ is equal to

$$z_{S'} = 0 + D'_{jn} + D'_{n,n-1} + \cdots + D'_{j+1,j-1} + \cdots + D'_{21} + T'_1.$$

If we re-arrange the inequality $z_{S'} \geq z_S$, we obtain $D'_{jn} + D'_{j+1,j-1} \geq D'_{j+1,j} + D'_{j,j-1}$. From Property 2, we have $D'_{j+1,j-1} \geq D'_{j+1,j} + D'_{j,j-1}$, so the inequality holds.

Case 2. $j = 1$: $z_{S'}$ is equal to

$$z_{S'} = 0 + D'_{1n} + D'_{n,n-1} + \cdots + D'_{32} + T'_2.$$

If we re-arrange the inequality $z_{S'} \geq z_S$, we obtain $D'_{1n} + T'_2 \geq D'_{21} + T'_1$. From Property 3, we have $T'_2 - T'_1 \geq D'_{21}$, which means the inequality holds.

We can generalize this result by investigating the cases where more than one job are inserted before job n. Assuming there are two jobs $j, k, \forall j, k \in J \setminus \{n\}$, to be inserted before job n, we construct the new tour S' with regard to three cases:

Case 1. $k > j > 1$: $z_{S'}$ is equal to

$$z_{S'} = 0 + D'_{jk} + D'_{kn} + D'_{n,n-1} + \cdots + D'_{j+1,j-1} + \cdots + D'_{j+1,j-1} + \cdots + D'_{21} + T'_1.$$

If we re-arrange the inequality $z_{S'} \geq z_S$, we obtain $D'_{jk} + D'_{kn} + D'_{k+1,k-1} + D'_{j+1,j-1} \geq D'_{j+1,j} + D'_{j,j-1} + D'_{k+1,k} + D'_{k,k-1}$. From Property 2, we have $D'_{j+1,j-1} \geq D'_{j+1,j} + D'_{j,j-1}$ and $D'_{k+1,k-1} \geq D'_{k+1,k} + D'_{k,k-1}$, so the inequality holds.

Case 2. $k > j = 1$: $z_{S'}$ is equal to

$$z_{S'} = 0 + D'_{1k} + D'_{kn} + D'_{n,n-1} + \cdots + D'_{k+1,k-1} + \cdots + D'_{32} + T'_2.$$

If we re-arrange the inequality $z_{S'} \geq z_S$, we obtain $D'_{1k} + D'_{kn} + D'_{k+1,k-1} + T'_2 \geq D'_{k+1,k} + D'_{k,k-1} + D'_{21} + T'_1$. From Property 3, we have $T'_2 - T'_1 \geq D'_{21}$ and from Property 2 we have $D'_{k+1,k-1} \geq D'_{k+1,k} + D'_{k,k-1}$, which mean the inequality holds.

Case 3. $k = 2, j = 1$: $z_{S'}$ is equal to

$$z_{S'} = 0 + D'_{12} + D'_{2n} + D'_{n,n-1} + \cdots + D'_{43} + T'_3.$$

If we re-arrange the inequality $z_{S'} \geq z_S$, we obtain $D'_{12} + D'_{2n} + T'_3 \geq D'_{32} + D'_{21} + T'_1$. From the generalization of Property 3, we have $T'_3 - T'_1 \geq D'_{31}$ and from Property 2 we have $D'_{31} \geq D'_{32} + D'_{21}$, which mean the inequality holds.

Inserting more than two jobs will also result in the same arguments. This completes the proof. □

We note that the SPT and LPT sequences can be obtained in $O(n \log n)$ time. The following theorem shows that the SPT sequence minimizes the makespan for the coupled task ordered flow-shop scheduling problem if the largest job is processed on the last machine.

Theorem 6. *The minimum makespan for the coupled task ordered flow-shop scheduling problem is obtained by the SPT sequence if M_m^{max}.*

Proof. We claim the proof similar to those presented in Panwalkar and Woollam (1979) and Smith et al. (1975), and by noting that the case with M_m^{max} is the reverse of the case with M_1^{max}, where the route for executing the jobs is reversed. From the reverse property of scheduling problems (McMahon & Burton, 1967), we see that the reverse of an optimal sequence of the case M_1^{max} obtained by the LPT rule is the SPT sequence, implying that the largest job is processed on the last machine. □

Consider the problem $I_{3\times3}$. Observe that the mirror image of the LPT sequence depicted in Fig. 3 is the SPT sequence, which is optimal, and no job can start earlier.

5. Concluding remarks

We explored several optimal properties of the coupled task flow-shop scheduling problem to minimize the makespan. With arbitrary values for the delays, we transformed the two-machine case with permutation schedules into the equivalent no-wait case. We showed that the problem is polynomially solvable if there are only two machines, and is strongly NP-hard if there are more than two machines. Our result complements the existing result that the coupled task flow-shop scheduling problem with non-permutation schedules is strongly NP-hard even for the two-machine case. We introduced the additional assumption of ordered delays, which may arise in practice. We proved

that under the ordered delay assumption, an optimal sequence follows the pyramidal structure. We proposed a polynomial dynamic program to solve the case. We further showed that if the largest task of every job occurs on the first or the last machine, the LPT or SPT sequence is optimal, respectively. Future research on this problem may consider objective functions other than the makespan. In addition, the problem settings that are computationally intractable, e.g., the non-permutation case may not be solved by our methods. Solving the non-permutation setting may warrant approximate algorithms.

CRediT authorship contribution statement

Mostafa Khatami: Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **Amir Salehipour:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **T.C.E. Cheng:** Writing – review & editing.

Data availability

No data was used for the research described in the article.

Acknowledgments

We thank the editors and anonymous referees for their valuable and constructive comments on earlier versions of the paper. Mostafa Khatami is a recipient of UTS International Research Scholarship (IRS) and UTS President's Scholarship (UTSP). Amir Salehipour is a recipient of the Australian Research Council Discovery Early Career Researcher Award (project number DE170100234) funded by the Australian Government. Cheng was also 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., & Baburin, A. E. (2007). Approximation algorithms for UET scheduling problems with exact delays. *Operations Research Letters*, *35*(4), 533–540.

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

Arora, R. K., & Rana, S. P. (1980). Scheduling in a semi-ordered flow-shop without intermediate queues. *AIIE Transactions*, *12*(3), 263–272.

Azadeh, A., Farahani, M. H., Torabzadeh, S., & Baghersad, M. (2014). Scheduling prioritized patients in emergency department laboratories. *Computer Methods and Programs in Biomedicine*, *117*(2), 61–70.

Baker, K. R., & Trietsch, D. (2013). *Principles of sequencing and scheduling*. John Wiley & Sons.

Bikker, I. A., Kortbeek, N., van Os, R. M., & Boucherie, R. J. (2015). Reducing access times for radiation treatment by aligning the doctor's schemes. *Operations Research for Health Care*, *7*, 111–121.

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

Burkard, R. E., Deineko, V. G., van Dal, R., van der Veen, J. A. A., & Woeginger, G. J. (1998). Well-solvable special cases of the traveling salesman problem: a survey. *SIAM Review*, *40*(3), 496–546.

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

Emmons, H., & Vairaktarakis, G. (2012). *Flow shop scheduling: Theoretical results, algorithms, and applications, Vol. 182*. Springer Science & Business Media.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, *1*(2), 117–129.

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

Hejazi, S. R., & Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: a review. *International Journal of Production Research*, *43*(14), 2895–2929.

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

Khatami, M., & Salehipour, A. (2021). Coupled task scheduling with time-dependent processing times. *Journal of Scheduling*, *24*(2), 223–236.

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

Khatami, M., Salehipour, A., & Hwang, F. J. (2019). Makespan minimization for the *m*-machine ordered flow shop scheduling problem. *Computers & Operations Research*, *111*, 400–414.

Lehoux-Lebacque, V., Brauner, N., & 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., & Zhao, H. (2007). Scheduling two-machine flow shops with exact delays. *International Journal of Foundations of Computer Science*, *18*(02), 341–359.

Marinagi, C. C., Spyropoulos, C. D., Papatheodorou, C., & Kokkotos, S. (2000). Continual planning and scheduling for managing patient tests in hospital laboratories. *Artificial Intelligence in Medicine*, *20*(2), 139–154.

McMahon, G., & Burton, P. (1967). Flow-shop scheduling with the branch-and-bound method. *Operations Research*, *15*(3), 473–481.

Panwalkar, S. S., & Woollam, C. R. (1979). Flow shop scheduling problems with no in-process waiting: A special case. *Journal of the Operational Research Society*, *30*(7), 661–664.

Pérez, E., Ntaimo, L., Malavé, C. O., Bailey, C., & McCormack, P. (2013). Stochastic online appointment scheduling of multi-step sequential procedures in nuclear medicine. *Health Care Management Science*, *16*(4), 281–299.

Pérez, E., Ntaimo, L., Wilhelm, W. E., Bailey, C., & McCormack, P. (2011). Patient and resource scheduling of multi-step medical procedures in nuclear medicine. *IIE Transactions on Healthcare Systems Engineering*, *1*(3), 168–184.

Röck, H. (1984). The three-machine no-wait flow shop is NP-complete. *Journal of the ACM*, *31*(2), 336–345.

Rossit, D. A., Tohmé, F., & Frutos, M. (2018). The non-permutation flow-shop scheduling problem: a literature review. *Omega*, *77*, 143–153.

Saure, A., Patrick, J., Tyldesley, S., & Puterman, M. L. (2012). Dynamic multi-appointment patient scheduling for radiation therapy. *European Journal of Operational Research*, *223*(2), 573–584.

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

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

Smith, M. L. (1968). *A critical analysis of flow-shop sequencing* (Ph.D. thesis), Texas Tech University.

Smith, M. L., Panwalkar, S. S., & Dudek, R. A. (1975). Flowshop sequencing problem with ordered processing time matrices. *Management Science*, *21*(5), 544–549.

Smith, M. L., Panwalkar, S. S., & Dudek, R. A. (1976). Flowshop sequencing problem with ordered processing time matrices: A general case. *Naval Research Logistics Quarterly*, *23*(3), 481–486.

van der Veen, J. A. A., & van Dal, R. (1991). Solvable cases of the no-wait flow-shop scheduling problem. *Journal of the Operational Research Society*, *42*(11), 971–980.

Yu, W., Hoogeveen, H., & Lenstra, J. K. (2004). Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard. *Journal of Scheduling*, *7*(5), 333–348.