Annals of Operations Research
https://doi.org/10.1007/s10479-024-06275-z

ORIGINAL RESEARCH

Two-agent proportionate flowshop scheduling with deadlines: polynomial-time optimization algorithms

Kuo-Ching Ying[1] · Pourya Pourhejazy[2] · Chuan-En Sung[1,3]

Received: 22 December 2021 / Accepted: 4 September 2024

Abstract
Volatility in the supply chain of critical products, notably the vaccine shortage during the pandemic, influences livelihoods and may lead to significant delays and long waiting times. Considering the capital- and time-intensive nature of capacity expansion plans, strategic operational production decisions are required best to address the supply-demand mismatches given the limited production resources. This research article investigates the production scenarios where the demand of one agent must be completed within a specified due date, hereinafter referred to as the *deadline*, while minimizing the maximum or total completion time of another agent's demand. For this purpose, the Two-Agent Proportionate Flowshop Scheduling Problem with deadlines is introduced. Two polynomial-time optimization algorithms are developed to solve these optimization problems. This study will serve as a basis for further developing this practical yet understudied scheduling problem.

Keywords Short-term production planning · Multi-agent scheduling · Conflicting tasks · Polynomial-time optimization

List of symbols

n^k	The number of jobs associated with agent k.
m	The number of available machines.
j	Job index, $j = 1, 2, ..., n^k$
k	Agent index, which is $k = A, B$ for dual-agent situations.
i	Machine index, $i = 1, 2, ..., m$
g	Possible job positions in a sequence, $g = 1, ..., n$.
g^k	Possible positions of the jobs associated with agent k, $g^k = 1, ..., n^k$.

✉ Pourya Pourhejazy
pourya.pourhejazy@uit.no

[1] Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 106, Taiwan

[2] Department of Industrial Engineering, UiT- The Arctic University of Norway, Lodve Langesgate 2, 8514 Narvik, Norway

[3] Powertech Technology Inc., No. 10, Datong Rd., Hukou Township, Hsinchu, 303035, Taiwan

$J_{[j]}^{k}$	Job situated in position j of the sequence associated with agent k.
F_k	Job cluster associated with agent k, $\mathrm{F}_k = \left\{\mathrm{J}_1^k, \mathrm{J}_2^k, \ldots, \mathrm{J}_{n^k}^k\right\}$.
$p_{[j]}^{k}$	Processing time of j th job in the sequence associated with agent k.
$d_{[j]}^{A}$	Due date of j th job in the sequence associated with agent k.
$C_{j,i}$	Integer decision variable; completion time of job j on machine i.
$C_{\max}^{B}$	The makespan of the jobs associated with Agent B.
$\sum C_j^B$	The total completion time of the jobs associated with Agent B

1 Introduction

In flowshop scheduling, all jobs go through an identical sequence of machines/processes. In certain application areas, such as the painting process in the automotive industry and modern bio-medicine production processes, the processing time of the jobs on the machines is about the same; this particular variant of flowshop scheduling is known as the Proportionate flowshop scheduling problem (PFSP). Choi et al. (2010) argued that minimizing the maximum completion time (makespan) in PFSP with machine-dependent processing times is *NP*-hard. Other studies developed exact solution algorithms for solving PFSPs with other objective functions, such as maximum earliness (Mor and Mosheiov 2015a), the number of early jobs (Mor and Mosheiov 2015b), minsum, and minmax (Mor and Mosheiov 2016), total absolute deviation of job completion times (Kovalev et al. 2019), and total completion time (Hertrich et al. 2020).

Different extensions to the PFSPs have been investigated in the scheduling literature to address practical production requirements. No-wait PFSP (Gerstl et al. 2015), permutation PFSPs (Cheng et al. 2018), PFSP with job rejection (Agnetis et al. 2017), unequal machine speeds (Panwalkar and Koulamas 2017), controllable processing times (Oron 2019), with position-dependent weights (Jiang et al. 2019), due date constraints (Sun et al. 2020), due window assignment (Qian and Han 2022); job rejection and common due date assignment (Geng et al. 2023) are some seminal examples. For a detailed elaboration on the PFSPs extensions, we refer interested readers to the systematic literature review by Panwalkar et al. (2013). The multi-agent PFSP has attracted relatively limited attention despite its relevance in addressing conflicting scheduling tasks in tight supply-demand situations.

The early multi-agent scheduling studies investigated the problem in single-machine production environments (see Ng et al. (2006)). Agnetis et al. (2004) were one of the first groups to extend the problem to work in a more realistic production situation with multiple identical machines, which was proven to be an *NP*-hard problem in a strong sense when considering the makespan (Leung et al. 2010). Among the most relevant studies, Mor and Mosheiov (2014) developed a polynomial-time solution algorithm to solve three variants of the Two-Agent Proportionate Flowshop Scheduling Problems (TAPFSP) where a maximum cost is considered for one agent while the total cost, completion time, and delay times of the other agent are minimized, respectively. Gerstl et al. (2019) developed a pseudo-polynomial-time dynamic programming algorithm to solve TAPFSP while considering total late work as the optimization criterion for the second agent. Most recently, Chen and Li (2021) developed a pseudo-polynomial-time algorithm to solve TAPFSP while minimizing the sum of the weighted late works of one agent and the weighted number of late jobs of the other agent. They also developed a polynomial-time optimization algorithm to solve TAPFSP.

In reality, there are situations when the supply chain of certain critical products is under pressure, and there is an urgency to satisfy new demands while ensuring that existing ones are completed within a certain due date. Taking the COVID-19 vaccine shortage in the early stages of the pandemic as an example, a pharmaceutical company may receive new orders while they have an obligation to fulfill the orders placed in advance with specific requirements. In this case, the production planner is interested in minimizing the time required to fulfill the new demands (makespan) or maximizing the utilization of the production resources (total completion time) (Pourhejazy 2024) while keeping in mind that there is a due date for earlier commitments. This practical setting has not been sufficiently investigated in the literature on TAPFSPs. The present study contributes to this understudied scheduling topic by developing two polynomial-time optimization algorithms to exactly solve the TAPFSP with deadlines, considering the total completion time and makespan.

The TAPFSP with deadlines investigated in this study is hereafter denoted by $\mathrm{P}F_m \mid \mathrm{C}_j^A \leq d_j^A \mid \sum \mathrm{C}_j^B$ and $\mathrm{P}F_m \mid \mathrm{C}_j^A \leq d_j^A \mid \mathrm{C}_{\max}^B$. $\mathrm{P}F_m$ refers to the proportionate flowshop setting considering m machines. $\mathrm{C}_j^A \leq d_j^A$ specifies that the completion time of job j from Agent A should be before the specified due date. Finally, $\mathrm{C}_{\max}^B$ and $\sum \mathrm{C}_j^B$ denotes the objective function, which are to minimize the maximum and total completion times of the jobs associated with Agent B, respectively.

The remainder of this research article begins with a comprehensive review of the PFSP literature in Section 2. The proposed polynomial-time optimization algorithms are presented in Section 3. Finally, the concluding remarks in Section 4 summarize the major findings and provide suggestions for future developments in the field.

2 Literature review

The PFSP was first introduced by Chin and Tsai (1981). Later, Ow (1985) solved the problem while considering the total delay time and found a near-optimum solution for the instances with three machines using Branch-and-Bound. Adenso-Díaz (1992, 1996) addressed the same problem by developing improved Tabu Search algorithms to obtain better competitive solutions. Allahverdi (1996) introduced the longest processing time (LPT) sequencing rule to minimize maximum lateness in a two-machine PFSP with breakdowns. Shakhlevich et al. (1998) developed a polynomial-time solution algorithm to solve the basic PFSP considering the total weighted completion time. Edwin Cheng and Shakhlevich (1999) developed a polynomial-time solution algorithm to address the bicriteria PFSP with controllable processing times. Allahverdi and Savsar (2001) introduced the shortest processing time (SPT) and LPT sequencing rules to minimize makespan in a two-machine PFSP with setup times and stochastic operational parameters. Ageev (2007) developed an approximation algorithm to deal with PFSPs with minimum delays and makespan criteria. Koulamas and Kyparisis (2007) proposed the SPT dispatching rule to exactly solve the PFSP problem in single- and two-machine factories, minimizing the total completion time. Shiau et al. (2008) proposed the proportionate flexible flowshop scheduling problem and used a Genetic Algorithm to find a near-optimal solution, considering the minimized total weighted completion time. Huang and Shiau (2008) and Dong et al. (2015) developed column-generation and polynomial-time approximation algorithms to solve the same problem.

More recent studies addressed more complex PFSPs with new constraints and decision variables. Koulamas and Kyparisis (2009) introduced PFSPs with the bottleneck machine.

Mosheiov and Oron (2012) developed a polynomial-time solution algorithm to minimize the number of tardy jobs in PFSPs with position-dependent processing times, in which processing times vary considering the number of previously processed jobs. Mor and Mosheiov (2015b) developed an iterative search algorithm and showed that the PFSP is polynomial-time solvable when the number of early jobs is considered as the optimization criterion. They showed in later research that the problem is also polynomial-time solvable when considering maximum earliness (Mor and Mosheiov 2015a). Panwalkar and Koulamas (2015a) introduced the PFSP with missing operations. The same authors developed a polynomial-time solution algorithm to solve PFSPs with fixed job processing times, position-dependent job processing times, controllable job processing times, as well as a variant with job rejection (Panwalkar and Koulamas 2015b). Mor and Mosheiov (2016) introduced the PFSP with common flow allowance; they considered min-sum and min-max criteria to solve the problem using a polynomial-time solution algorithm. Shabtay and Oron (2016) developed exact and approximation algorithms for solving the PFSP with machine-dependent processing times and job rejection opportunities while considering various objective functions. Li et al. (2017) developed a pseudo-polynomial-time algorithm to solve the PFSP with job rejection, considering maximum tardiness and total weighted completion time.

The above studies considered equal machine speeds, which impact the complexity of the problem. Hou and Hoogeveen (2003) were the first to develop a polynomial-time solution algorithm for a three-machine PFSP with unequal machine speeds, minimizing the makespan. Later, Choi et al. (2006) and Panwalkar and Koulamas (2017) developed heuristic algorithms to address two-machine PFSP, considering total weighted completion time, and makespan, respectively, for unequal machine speeds.

In another special case of flowshop scheduling with implications for problem complexity, Gerstl et al. (2015) developed a polynomial-time solution algorithm to exactly solve the no-wait variant of PFSP for minimizing the weighted number of just-in-time jobs. Ben-Yehoshua et al. (2015) adapted the LPT rule to address the no-wait PFSP with two operating machines. They showed that the problem remains exactly solvable for up to nine jobs while considering the total absolute deviation of job completion times. Kovalev et al. (2019) also developed a pseudo-polynomial-time dynamic programming algorithm to solve the no-wait PFSPs in a two-machine setting, aiming at minimizing the total absolute deviation of job completion time.

Among the most recent studies, Cheng et al. (2018) developed a polynomial-time solution algorithm to address the permutation variant of PFSP with variable maintenance works and investigated several objective functions, including total completion time, maximum lateness, and maximum tardiness. Gerstl et al. (2019) developed a pseudo-polynomial-time dynamic programming algorithm to minimize PFSPs while considering total late work. Mor and Shapira (2019) developed a similar optimization approach to address PFSPs with job rejection, considering the makespan criterion. Oron (2019) introduced the PFSP with batch processing, which was solved using a polynomial-time solution algorithm considering the makespan. Pseudo-polynomial-time dynamic programming algorithms were also used for solving PFSP with job rejection by Mor and Shapira (2020), where a constraint was included to limit the total rejection cost. Koulamas (2020) developed the same approach for solving a PFSP considering total tardiness.

The PFSP with a due date constraint was first introduced by Sun et al. (2020), who developed a polynomial-time solution algorithm to solve the problem considering the total weighted cost. PFSP with release dates and batch processing was solved exactly by Hertrich et al. (2020). Most recently, Lv and Wang (2021) developed a polynomial-time solution algorithm to solve the PFSP with due dates and position-dependent weights to minimize the makespan. Mor et al. (2021) developed a pseudo-polynomial-time dynamic programming approach to solve the PFSP with job rejection, considering total tardiness and rejection costs. All the above variants of PFSP are suitable for normal operating situations, where production planning and scheduling are done for all jobs at hand. However, there are situations where the maximum/total completion time of new demands should be minimized while ensuring that the current demand is fulfilled within a certain due date. In this situation, the limited available production resources, i.e., machines, need to be considered for fulfilling two different sets of requirements. Such a practical situation can be addressed using two-agent scheduling. Multi-agent PFSPs have been limited to a few studies. Estévez-Fernández et al. (2008) were the first to develop a game model to address multi-agent PFSPs. Later, Mor and Mosheiov (2014) developed a polynomial-time solution approach based on the Lawler algorithm to solve the two-agent PFSP. In their study, the maximum cost of all the jobs, total completion time, and the minimum number of tardy jobs were considered for the first agent, and an upper bound on the maximum allowable cost was considered for the second agent. Li et al. (2018) developed a pseudo-polynomial-time dynamic programming algorithm to solve the multi-agent PFSP, where every agent attempts to maximize its total gains of just-in-time jobs.

Most recently, Chen and Li (2021) developed a pseudo-polynomial-time solution algorithm to solve the two-agent variant of PFSP while considering machine-dependent processing time. In their model, the scheduling of the first agent is done to minimize the total weighted late work, while the total weighted number of late jobs for the second agent is minimized. The following section develops two polynomial-time optimization algorithms to solve the TAPFSP with deadlines, which is particularly practical for addressing the production scheduling of critical products during times of shortage.

3 Proposed method

Let's assume a production site with limited capacity equipped with $M = \{1, 2, ..., m\}$ machinery, which must process jobs from k agents, i.e., n^k. Assume a dual-agent problem, a total of $n = n^A + n^B$ jobs are involved. Jobs follow the same routine, and the processing time of each job is the same across all production processes (machines). It is assumed that each machine $i = 1, 2, ..., m$ can process only one job $j = 1, 2, ..., n^k$ from one of the agents at any given time. Once the processing of a job on a machine is started, it cannot be assigned to another machine. The processing times of jobs are sequence-independent and uncorrelated. Additionally, the processing time and due date are assumed to be deterministic. Finally, we assume that the machines' downtime and defect rate are negligible, and scrap or rework are not allowed. The following notations are defined to establish two polynomial-time optimization algorithms.

The following sub-sections elaborate on the TAPFSP with deadlines, considering the total completion time and the makespan criterion.

3.1 Maximum completion time (Makespan), $PF_m \mid C_j^A \le d_j^A \mid C_{\max}^B$

Lemma 1 *In the case that the job sequence associated with Agent B is regarded as a group, the arbitrary arrangement of the jobs does not impact the maximum completion time.*

Proof See (Chin and Tsai 1981). Therefore, the jobs associated with one agent can be arbitrarily sorted without affecting their maximum completion time.

Theorem 1 *Given that jobs associated with Agent B as one processing group, there is a set of optimal solutions to the problem if the job sequence of Agent A is sorted according to EDD.*

Proof

- Assume that the jobs associated with Agent B are treated as one group. In so doing, the initial sequence is $\pi_1 = \left(\ldots, J_j^B, J_{j+1}^B, J_{j'}^A, J_{j'+1}^A, \ldots\right)$ and it is assumed that $C_{j'+1}^A > d_{j'+1}^A$. Considering Figure 1 as an illustrative example, job $J_{j'+1}^A$ of sequence π_1 is sorted as a group and the jobs associated with Agent B are considered as non-group, denoted by π_2, with its sorted job sequence being shown by π_2'. It can be observed that the makespan of Agent B under both π_2 and π_2' are equal, i.e., $C_{\max}^B(\pi_2) = C_{\max}^B(\pi_2')$. In this situation, it can be considered that using the job sequence of Agent B as a group minimizes the completion time of Agent A under the condition that the makespan of Agent B remains unchanged

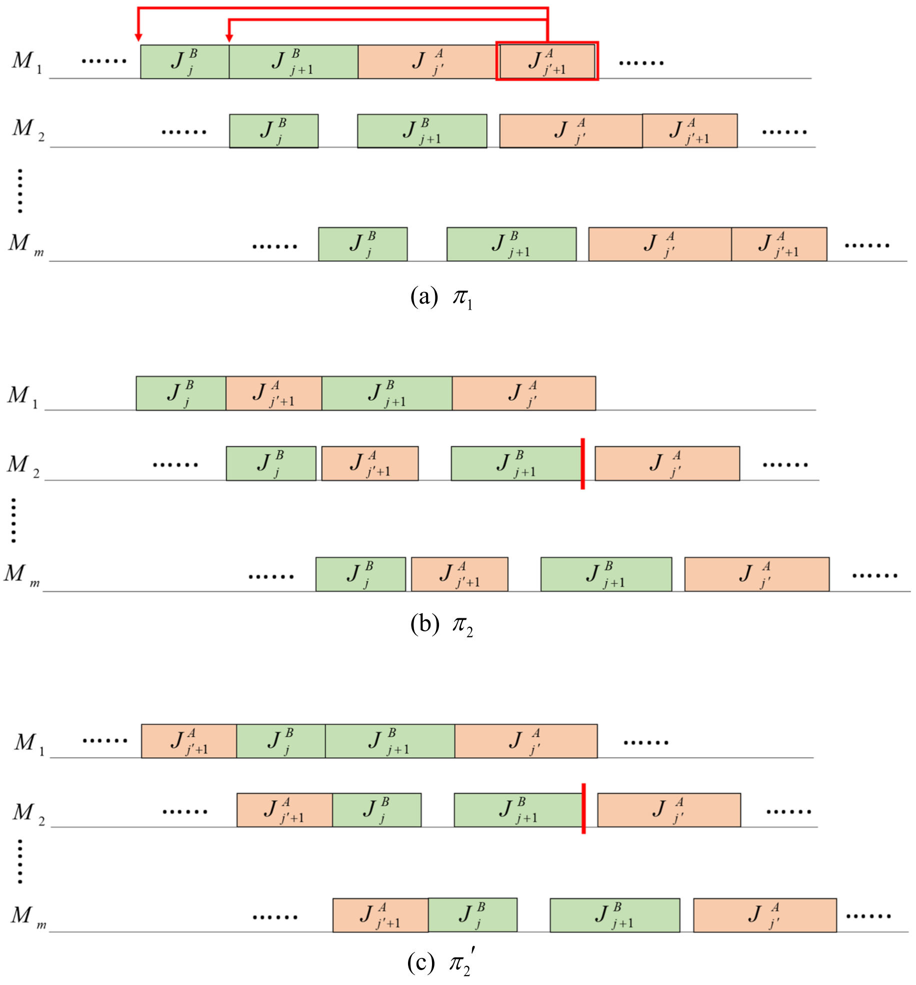

Fig. 1 An illustrative example for Theorem 1

and the chance of Agent A exceeding the due date is also reduced. Therefore, it can be proven that a better solution can be obtained by sorting the jobs associated with Agent B as a group in the problem of $\left|PF_m\right|C_j^A \le d_j^A\left|C_{\max}^B\right.$

- Sort Agent A's jobs according to EDD. As shown in Figure 2, assuming Agent B's job sequence as a group and that $\mathrm{d}_{j'}^A < d_{j'+1}^A$, we compare the situations where Agent A's job is organized using non-EDD rule, i.e., $\pi_1{=}\left(J_j^B, J_{j+1}^B, J_{j'+1}^A, J_{j'}^A\right)$, and EDD, i.e., $\pi_1'{=}\left(\mathrm{J}_j^B, J_{j+1}^B, J_{j'}^A, J_{j'+1}^A\right)$. In this situation, it can be observed that $\mathrm{C}_{j'}^A > d_{j'}^A$ and $\mathrm{C}_{j'+1}^A > d_{j'+1}^A$. Then we move $J_{j'+1}^A$ of π_1 of and $J_{j'}^A$ of π_1' to the front to get π_2 (Figure 2(c)) and π_2' (Figure 2(d)). It can be observed that $\mathrm{C}_{\max}^B(\pi_2) = \mathrm{C}_{\max}^B(\pi_2')$ and $\mathrm{C}_{j'}^A(\pi_2) > d_{j'}^A$. In this situation, the completion time of Agent A, i.e., $C_{j'}^A(\pi_2), C_{j'+1}^A(\pi_2), C_{j'}^A(\pi_2'), C_{j'+1}^A(\pi_2')$ is calculated as shown in Equation (1).

$$
\begin{aligned}
C_{j'}^A(\pi_2) &= \left(p_{[1]}^A + p_{[2]}^A + \dots + p_{j'-1}^A\right) + \left(p_{[1]}^B + p_{[2]}^B + \dots + p_{j-1}^B\right) + p_{j'}^A + p_{j'+1}^A \\
&\quad +(m-1)\max(p_{[1]}^A, \dots, p_{j'+1}^A, p_{[1]}^B, \dots, p_{j-1}^B) \\
C_{j'+1}^A(\pi_2) &= \left(p_{[1]}^A + p_{[2]}^A + \dots + p_{j'-1}^A\right) + \left(p_{[1]}^B + p_{[2]}^B + \dots + p_{j-1}^B\right) + p_{j'+1}^A \\
&\quad +(m-1)\max(p_{[1]}^A, \dots, p_{j'-1}^A, p_{j'+1}^A, p_{[1]}^B, \dots, p_{j-1}^B) \\
C_{j'}^A(\pi_2') &= \left(p_{[1]}^A + p_{[2]}^A + \dots + p_{j'-1}^A\right) + \left(p_{[1]}^B + p_{[2]}^B + \dots + p_{j-1}^B\right) + p_{j'}^A \\
&\quad +(m-1)\max(p_{[1]}^A, \dots, p_{j'}^A, p_{[1]}^B, \dots, p_{j-1}^B) \\
C_{j'+1}^A(\pi_2') &= \left(p_{[1]}^A + p_{[2]}^A + \dots + p_{j'-1}^A\right) + \left(p_{[1]}^B + p_{[2]}^B + \dots + p_{j-1}^B\right) + p_{j'}^A + p_{j'+1}^A \\
&\quad +(m-1)\max(p_{[1]}^A, \dots, p_{j'+1}^A, p_{[1]}^B, \dots, p_{j-1}^B)
\end{aligned}
\tag{1}
$$

Considering that $\mathrm{d}_{j'}^A < d_{j'+1}^A$, when the job sequence of Agent A is sorted based on the non-EDD rule, i.e., π_2: $C_{j'}^A(\pi_2) < d_{j'}^A$ and $C_{j'+1}^A(\pi_2) < d_{j'+1}^A$, which results in π_2': $C_j^A(\pi_2') < d_{j'}^A$ and $C_{j+1}^A(\pi_2') < d_{j'}^A$. On the other hand, following the EDD rule, π_2': $C_j^A(\pi_2') < d_{j'}^A$ and $C_{j+1}^A(\pi_2') < d_{j'}^A$. In this situation, for the non-EDD sequence π_2 , $C_{j'}^A(\pi_2) < d_{j'}^A$ and $C_{j'+1}^A(\pi_2) < d_{j'+1}^A$ may not be true. Therefore, it can be shown that a better solution can be obtained by arranging Agent A's job sequence with EDD resulting in the optimal outcomes.

Finally, the makespan of TAPFSPs can be calculated by $C_{\max} = \sum_{j=1}^{n} p_j{+}(m-1)max(p_1, \dots, p_n)$ as suggested by (Chin and Tsai 1981).

Moving Agent A's delayed jobs ahead of Agent B's job group will not affect the completion times of Agent A's other jobs that have not been moved. Therefore, based on Lemma 1 and Theorem 1, we can obtain an optimal solution for the $\mathrm{P}F_m \mid \mathrm{C}_j^A \le d_j^A \mid \mathrm{C}_{\max}^B$ problem using the following polynomial-time optimization algorithm.

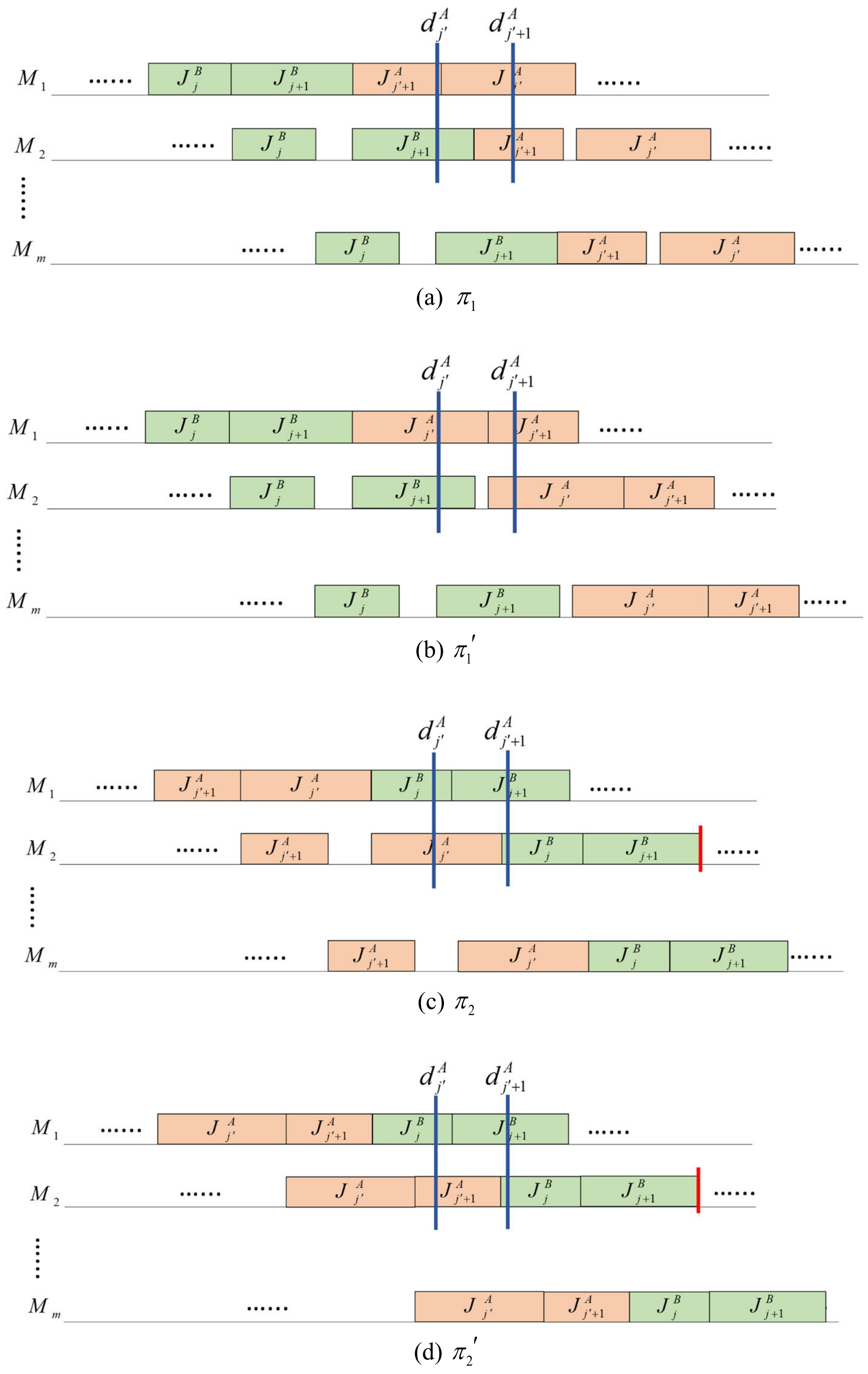

Fig. 2 An illustrative example for sorting Agent A's jobs according to EDD.

Algorithm 1

(1) Initialization Phase: First, sequence all the jobs of Agent B in an arbitrary order, and then arrange the jobs of Agent A after the last job of Agent B according to the EDD rule. The resulting job sequence is optimal if all of Agent A's jobs are completed before their due dates.
(2) Improvement Phase: If some of Agent A's jobs exceed their due dates, move Agent B's job group after the last delayed job of Agent A to generate the optimal solution. The resulting job sequence is optimal if all of Agent A's jobs are completed before their due dates. Otherwise, there is no feasible solution.

Theorem 2 *The computational complexity of Algorithm 1 for solving the* $PF_m \mid C_j^A \le d_j^A \mid C_{\max}^B$ *problem is* $O(n^A \log n^A)$

Proof The initialization phase of Algorithm 1 is performed in $O(n^A \log n^A)$, finding the last delayed job of Agent A requires $O(n^A)$, and moving the jobs of Agent B is done in constant time. Thus, the entire running time of the computational procedure of Algorithm 1 is $O(n^A \log n^A)$.

Illustrative Example A in Table 1 is used to better elaborate on the computational procedure of Algorithm 1.

I. Initialization Phase: In the illustrative example presented in Figure 3(a), three machines are considered for scheduling, where the makespan of Agent B's job should be minimized while ensuring that all of Agent A's jobs are completed before the specified due date. Considering Lemma 1 and Theorem 1, the order of jobs in Agent B's job group does not influence the maximum completion time; hence, they can be sorted arbitrarily. Subsequently, Agent A's jobs are sorted using the EDD rule. The resulting job sequence is $(J_1^B, J_2^B, J_3^B, J_3^A, J_1^A, J_2^A)$, where $C_3^A > d_3^A, C_1^A > d_1^A$.
II. Improvement Phase: The job group associated with Agent B is inserted after the last delayed job of Agent A, *i.e.*, J_1^A,, as shown in Figure 3(b). Since all of Agent A's jobs are completed before their due dates, the resulting job sequence $(j_3^A, j_1^A, j_1^B, j_2^B, j_3^B, j_2^A)$ is optimal.

Table 1 Parameters of the illustrative Example A

Agent	Job	Processing time	Due date
A	J_1^A	2	19
	J_2^A	3	25
	J_3^A	4	17
B	J_1^B	3	–
	J_2^B	2	–
	J_3^B	1	–

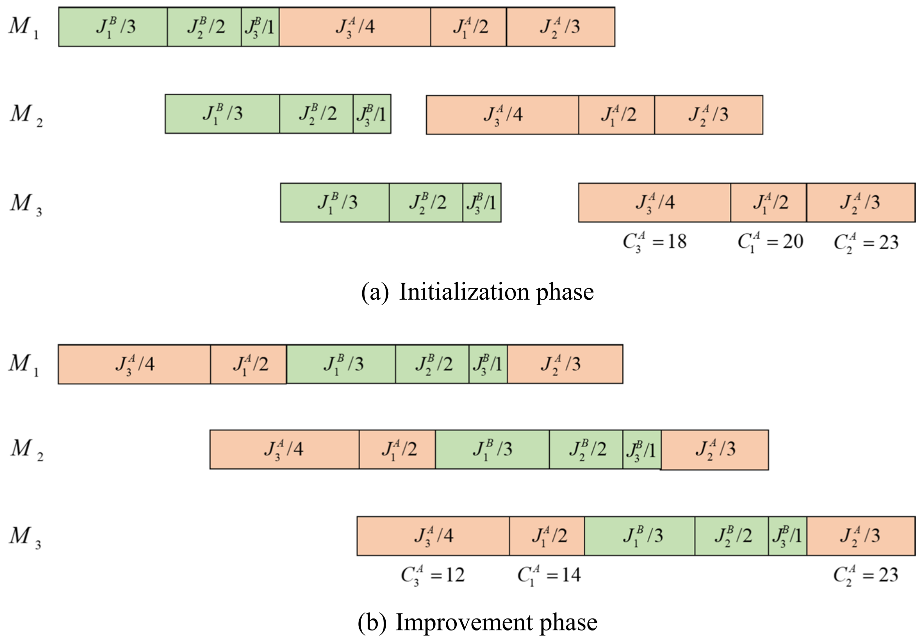

Fig. 3 An illustrative example of the computational procedure of Algorithm 1

3.2 *Total completion time,* $PF_m \mid C_j^A \le d_j^A \mid \sum C_j^B$

Theorem 3 *There is a set of optimal solutions for* $PF_m \mid C_j^A \le d_j^A \mid \sum C_j^B$ *when the jobs associated with Agent B are sorted using the SPT rule.*

Proof We know from the literature (Baker and Smith 2003) that SPT is the optimal approach for sorting jobs when solving the single-machine scheduling problem and the PFSP to minimize the total completion time. The PFSP with due dates and conflicting agents is comparatively more complex. Since our solution objective function is to minimize the total completion time of Agent B, we consider two approaches; first, we reduce the job sequence of Agent B considering the SPT rule, i.e., $\pi = (\ldots, J_j^B, J_{j'+1}^A, J_{j+1}^B, J_{j'}^A, \ldots)$, and non-SPT, i.e., $\pi' = (\ldots, J_{j+1}^B, J_{j'}^A, J_j^B, J_{j'+1}^A, \ldots)$; these are shown in Figure 4.

In this example, the total completion time of Agent B, considering $\pi = (J_j^B, J_{j'+1}^A, J_{j+1}^B, J_{j'}^A)$ and $\pi' = (J_{j+1}^B, J_{j'}^A, J_j^B, J_{j'+1}^A)$ sequences are calculated in Equations (2) and (3), respectively.

$$\begin{aligned}
&\sum C_j^B(\pi)\\
&= C_{[1]}^B + C_{[2]}^B + \ldots + C_j^B + C_{j+1}^B\\
&= C_{[1]}^B + C_{[2]}^B + \ldots + C_{j-1}^B +\\
&\left[\left(p_{[1]}^B + p_{[2]}^B + \ldots + p_j^B\right) + \left(p_{[1]}^A + p_{[2]}^A + \ldots + p_{j'-1}^A\right) + (m-1)\max\left(p_{[1]}^B, p_{[2]}^B, \ldots, p_j^B, p_{[1]}^A, p_{[2]}^A, \ldots, p_{j'-1}^A\right)\right]+\\
&\left[\left(p_{[1]}^B + p_{[2]}^B + \ldots + p_{j+1}^B\right) + \left(p_{[1]}^A + p_{[2]}^A + \ldots + p_{j'}^A\right) + (m-1)\max\left(p_{[1]}^B, p_{[2]}^B, \ldots, p_{j+1}^B, p_{[1]}^A, p_{[2]}^A, \ldots, p_{j'}^A\right)\right]
\end{aligned} \tag{2}$$

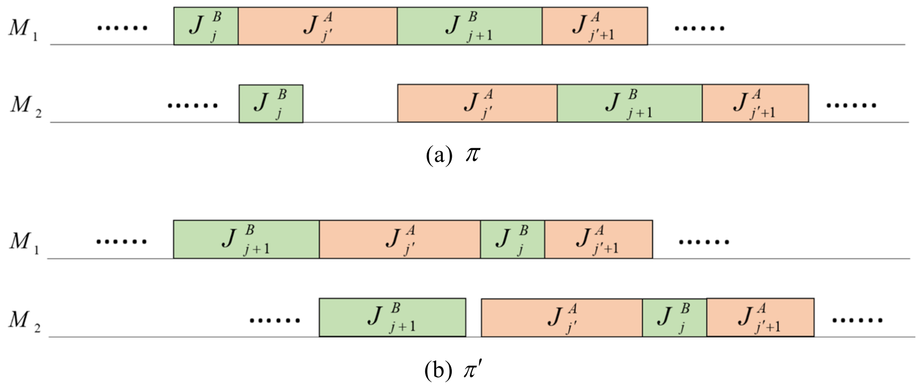

Fig. 4 An illustrative example of theorem 3

$$\begin{aligned}
&\sum C_j^B(\pi') \\
&= C_{[1]}^B + C_{[2]}^B + \ldots + C_j^B + C_{j+1}^B \\
&= C_{[1]}^B + C_{[2]}^B + \ldots + C_{j-1}^B + \\
&\left[\left(p_{[1]}^B + p_{[2]}^B + \ldots + p_{j+1}^B\right) + \left(p_{[1]}^A + p_{[2]}^A + \ldots + p_{j'}^A\right)\right. \\
&\left.+(m-1)\max\left(p_{[1]}^B, p_{[2]}^B, \ldots, p_{j+1}^B, p_{[1]}^A, p_{[2]}^A, \ldots, p_{j'}^A\right)\right] + \\
&\left[\left(p_{[1]}^B + p_{[2]}^B + \ldots + p_{j-1}^B + p_{j+1}^B\right) + \left(p_{[1]}^A + p_{[2]}^A + \ldots + p_{j'-1}^A\right)\right. \\
&\left.+(m-1)\max\left(p_{[1]}^B, p_{[2]}^B, \ldots, p_{j-1}^B, p_{j+1}^B, p_{[1]}^A, p_{[2]}^A, \ldots, p_{j'-1}^A\right)\right]
\end{aligned} \tag{3}$$

Given $p_j^B < p_{j+1}^B$, we can conclude that $\sum C_j^B(\pi) < \sum C_j^B(\pi')$, hence, a better solution is obtained for Agent B using the SPT approach.

In an optimal solution of the $PF_m \mid C_j^A \le d_j^A \mid \sum C_j^B$ problem, Agent B's jobs must be sorted according to the SPT rule, and Agent A's jobs must be sorted according to the EDD rule. Inserting Agent B's jobs before Agent A's delayed jobs does not alter the completion times of Agent A's delayed jobs. However, inserting one of Agent B's job immediately after the last delayed job of Agent A may decrease the completion time of Agent A's delayed jobs while minimally increasing the completion time of the inserted job of Agent B. Therefore, based on Theorems 1 and 3, we can quickly obtain an optimal solution for the $PF_m \mid C_j^A \le d_j^A \mid \sum C_j^B$ problem using the following polynomial-time optimization algorithm.

Algorithm 2

(1) Initialization Phase: First, sequence all the jobs of Agent B according to the SPT rule, and then arrange the jobs of Agent A after the last job of Agent B according to the EDD rule. The resulting job sequence is optimal if all of Agent A's jobs are completed before their due dates.
(2) Improvement Phase: If some of Agent A's jobs exceed their due dates, move the last job of Agent B to the first position after the last delayed job of Agent A. If there are still delayed jobs of Agent A, move the next-to-last job of Agent B to the first position after the last delayed job of Agent A. Continue this process until all of Agent A's jobs are completed before their due dates or until all of Agent B's jobs have been moved. The resulting job sequence is optimal if all of Agent A's jobs are completed before their due dates. Otherwise, there is no feasible solution.

Theorem 4 *The computational complexity of Algorithm 2 for solving the* $PF_m \mid C_j^A \le d_j^A \mid \sum C_j^B$ *problem is* $O(n^A n^B)$.

Proof The initialization phase of Algorithm 2 is performed in $O(n^B \log n^B + n^A \log n^A)$, finding the last delayed job of Agent A requires $O(n^A)$. There are at most n^B iterations to move the jobs of Agent B, in which each is performed in constant time. Thus, the entire running time of the computational procedure of Algorithm 2 is $O(n^A n^B)$.

Illustrative Example B in Table 2 is used to better elaborate on the computational procedure of Algorithm 2.

I. Initialization Phase: In the illustrative example presented in Figure 5(a), two machines are considered for scheduling, where the total completion time of Agent B should be minimized while ensuring that every job of Agent A is completed before the specified due date. First, sequence all the jobs of Agent B according to the SPT rule, and then arrange the jobs of Agent A after the last job of Agent B according to the EDD rule. The resulting job sequence is $(J_1^B, J_2^B, J_1^A, J_2^A)$, where $C_1^A > d_1^A$, $C_2^A > d_2^A$.

II. Improvement Phase: As shown in Figure 5(b), move the last job of Agent B, *i.e.*, J_2^B, to the first position after the last delayed job of Agent A, *i.e.*, J_2^A. The resulting job sequence is $(J_1^B, J_1^A, J_2^A, J_2^B)$, where $C_1^A > d_1^A$. Since there is still one delayed job of Agent A, move the next-to-last job of Agent B, *i.e.*, J_1^B, to the first position after the delayed job J_1^A, as shown in Figure 5(c). Since all of Agent A's jobs are completed before their due dates, the resulting job sequence $(J_1^A, J_1^B, J_2^A, J_2^B)$ is optimal.

Table 2 Parameters of the illustrative Example B

Agent	Job	Processing time	Due date
A	J_1^A	4	10
	J_2^A	2	16
B	J_1^B	4	–
	J_2^B	10	–

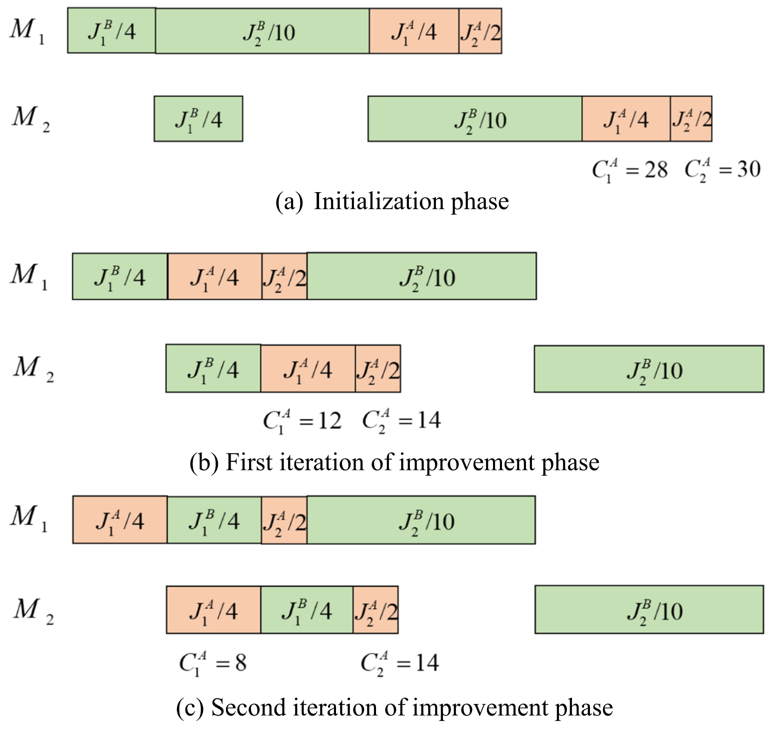

Fig. 5 An illustrative example of the computational procedure of Algorithm 2

4 Concluding remarks

Considering the limited available resources on a production site, companies often face the problem of satisfying new demands while ensuring the timely fulfillment of the existing ones. In this situation, proportionate scheduling helps maximize the outcomes while maintaining an acceptable level of responsiveness. This study investigated the TAPFSP with deadlines by developing two polynomial-time optimization algorithms to solve the problem with the objectives of minimizing the makespan and total completion time, respectively. Our findings can be considered a basis for further developments in the literature on proportionate flowshop scheduling.

This study is limited in that only two conflicting objectives are considered, while the real-world situation may require a more diverse set of demands. Future studies may build on our findings to address this limitation; for example, by applying our method to a multi-agent problem to check whether it can yield the optimal solution. The following suggestions may provide further insights into the future development of this relatively understudied scheduling extension. First, proportionate scheduling requires development in other scheduling environments, such as job-shop, open-shop, and parallel machines. The algorithms and lemmas developed in this study may be tested in the aforementioned scheduling situations. The second suggestion comes from integrating other practical features, such as no-idle and blocking constraints, to extend the real-world reach of the problem. Finally, revenue management-related decision variables can be included in the base model to better reflect the real-world situation. In so doing, one can include the possibility of rejecting orders or partially accepting them while considering the expected profit.

Acknowledgements We sincerely thank the anonymous reviewers for their valuable suggestions, which greatly improved the initial version of this research article; their feedback has been instrumental in refining our research and enhancing its overall quality.

Funding Open access funding provided by UiT The Arctic University of Norway (incl University Hospital of North Norway).

Declarations

Conflict of interest The authors declare that they have no known conflict of interest.

Ethical approval This article does not contain any studies with human participants performed by any of the authors.

References

Adenso-Díaz, B. (1992). Restricted neighborhood in the tabu search for the flowshop problem. *European Journal of Operational Research, 62*(1), 27–37. https://doi.org/10.1016/0377-2217(92)90174-8

Adenso-Díaz, B. (1996). An SA/TS mixture algorithm for the scheduling tardiness problem. *European Journal of Operational Research, 88*(3), 516–524. https://doi.org/10.1016/0377-2217(94)00213-4

Ageev, A. A. (2007). A 3/2-Approximation for the proportionate two-machine flow shop scheduling with minimum delays. In: *Approximation and Online Algorithms* Springer, Berlin 55–66

Agnetis, A., Mirchandani, P. B., Pacciarelli, D., & Pacifici, A. (2004). Scheduling problems with two competing agents. *Operations Research, 52*(2), 229–242. https://doi.org/10.1287/opre.1030.0092

Agnetis, A., & Mosheiov, G. (2017). Scheduling with job-rejection and position-dependent processing times on proportionate flowshops. *Optim Lett, 11*, 885–892. https://doi.org/10.1007/s11590-016-1059-8

Allahverdi, A. (1996). Two-machine proportionate flowshop scheduling with breakdowns to minimize maximum lateness. *Computers & Operations Research, 23*(10), 909–916. https://doi.org/10.1016/0305-0548(96)00012-3

Allahverdi, A., & Savsar, M. (2001). Stochastic proportionate flowshop scheduling with setups. *Computers & Industrial Engineering, 39*(3–4), 357–369. https://doi.org/10.1016/S0360-8352(01)00011-0

Baker, K. R., & Cole Smith, J. (2003). A multiple-criterion model for machine scheduling. *Journal of Scheduling, 6*(1), 7–16. https://doi.org/10.1023/A:1022231419049

Ben-Yehoshua, Y., Hariri, E., & Mosheiov, G. (2015). A note on minimising total absolute deviation of job completion times on a two-machine no-wait proportionate flowshop. *International Journal of Production Research, 53*(19), 5717–5724. https://doi.org/10.1080/00207543.2014.991843

Chen, R.-X., & Li, S.-S. (2021). Proportionate flow shop scheduling with two competing agents to minimize weighted late work and weighted number of late jobs. *Asia-Pacific Journal of Operational Research, 38*(02), 2050046. https://doi.org/10.1142/S0217595920500463

Cheng, C.-Y., Ying, K.-C., Chen, H.-H., & Lin, J.-X. (2018). Optimization algorithms for proportionate flowshop scheduling problems with variable maintenance activities. *Computers & Industrial Engineering, 117*, 164–170. https://doi.org/10.1016/j.cie.2018.01.027

Chin, F. Y., & Tsai, L.-L. (1981). On J-maximal and J-minimal flow-shop schedules. *Journal of the ACM (JACM), 28*(3), 462–476.

Choi, B.-C., Leung, J.Y.-T., & Pinedo, M. L. (2010). A note on makespan minimization in proportionate flow shops. *Information Processing Letters, 111*(2), 77–81. https://doi.org/10.1016/j.ipl.2010.10.016

Choi, B.-C., Yoon, S.-H., & Chung, S.-J. (2006). Minimizing the total weighted completion time in a two-machine proportionate flow shop with different machine speeds. *International Journal of Production Research, 44*(4), 715–728. https://doi.org/10.1080/00207540500268780

Dong, J., Jiang, Y., Zhang, A., Hu, J., & Luo, H. (2015). An approximation algorithm for proportionate scheduling in the two-stage hybrid flow shop. *Information Processing Letters, 115*(4), 475–480. https://doi.org/10.1016/j.ipl.2014.11.014

Edwin Cheng, T. C., & Shakhlevich, N. (1999). Proportionate flow shop with controllable processing times. *Journal of Scheduling, 2*(6), 253–265. https://doi.org/10.1002/(SICI)1099-1425(199911/12)2:6%3c253::AID-JOS30%3e3.0.CO;2-R

Estévez-Fernández, A., Mosquera, M. A., Borm, P., & Hamers, H. (2008). Proportionate flow shop games. *Journal of Scheduling, 11*(6), 433–447. https://doi.org/10.1007/s10951-008-0062-z

Geng, X. N., Sun, X., Wang, J., & Pan, L. (2023). Scheduling on proportionate flow shop with job rejection and common due date assignment. *Computers & Industrial Engineering, 181*, 109317. https://doi.org/10.1016/j.cie.2023.109317

Gerstl, E., Mor, B., & Mosheiov, G. (2015). A note: Maximizing the weighted number of just-in-time jobs on a proportionate flowshop. *Information Processing Letters, 115*(2), 159–162. https://doi.org/10.1016/j.ipl.2014.09.004

Gerstl, E., Mor, B., & Mosheiov, G. (2019). Scheduling on a proportionate flowshop to minimise total late work. *International Journal of Production Research, 57*(2), 531–543. https://doi.org/10.1080/00207543.2018.1456693

Hertrich, C., Weiß, C., Ackermann, H., Heydrich, S., & Krumke, S. O. (2020). Scheduling a proportionate flow shop of batching machines. *Journal of Scheduling, 23*(5), 575–593. https://doi.org/10.1007/s10951-020-00667-2

Hou, S., & Hoogeveen, H. (2003). The three-machine proportionate flow shop problem with unequal machine speeds. *Operations Research Letters, 31*(3), 225–231. https://doi.org/10.1016/S0167-6377(02)00232-8

Huang, Y.-M., & Shiau, D.-F. (2008). Combined column generation and constructive heuristic for a proportionate flexible flow shop scheduling. *The International Journal of Advanced Manufacturing Technology, 38*(7–8), 691–704. https://doi.org/10.1007/s00170-007-1130-9

Jiang, C., Zou, D., Bai, D., & Wang, J. B. (2019). Proportionate flowshop scheduling with position-dependent weights. *Engineering Optimization, 52*(1), 37–52. https://doi.org/10.1080/0305215X.2019.1573898

Koulamas, C. (2020). The proportionate flow shop total tardiness problem. *European Journal of Operational Research, 284*(2), 439–444. https://doi.org/10.1016/j.ejor.2020.01.002

Koulamas, C., & Kyparisis, G. J. (2007). Single-machine and two-machine flowshop scheduling with general learning functions. *European Journal of Operational Research, 178*(2), 402–407. https://doi.org/10.1016/j.ejor.2006.01.030

Koulamas, C., & Kyparisis, G. J. (2009). A note on the proportionate flow shop with a bottleneck machine. *European Journal of Operational Research, 193*(2), 644–645. https://doi.org/10.1016/j.ejor.2008.01.031

Kovalev, S., Kovalyov, M. Y., Mosheiov, G., & Gerstl, E. (2019). Semi-V-shape property for two-machine no-wait proportionate flow shop problem with TADC criterion. *International Journal of Production Research, 57*(2), 560–566. https://doi.org/10.1080/00207543.2018.1468097

Leung, J.Y.-T., Pinedo, M., & Wan, G. (2010). Competitive Two-Agent Scheduling and Its Applications. *Operations Research, 58*(2), 458–469. https://doi.org/10.1287/opre.1090.0744

Li, S.-S., Chen, R.-X., & Li, W.-J. (2018). Proportionate flow shop scheduling with multi-agents to maximize total gains of JIT jobs. *Arabian Journal for Science and Engineering, 43*(2), 969–978. https://doi.org/10.1007/s13369-017-2900-9

Li, S.-S., Qian, D.-L., & Chen, R.-X. (2017). Proportionate flow shop scheduling with rejection. *Asia-Pacific Journal of Operational Research, 34*(04), 1750015. https://doi.org/10.1142/S0217595917500154

Lv, D.-Y., & Wang, J.-B. (2021). Study on proportionate flowshop scheduling with due-date assignment and position-dependent weights. *Optimization Letters, 15*(6), 2311–2319. https://doi.org/10.1007/s11590-020-01670-4

Mor, B., & Mosheiov, G. (2014). Polynomial time solutions for scheduling problems on a proportionate flowshop with two competing agents. *Journal of the Operational Research Society, 65*(1), 151–157. https://doi.org/10.1057/jors.2013.9

Mor, B., & Mosheiov, G. (2015). A note: Minimizing maximum earliness on a proportionate flowshop. *Information Processing Letters, 115*(2), 253–255. https://doi.org/10.1016/j.ipl.2014.09.023

Mor, B., & Mosheiov, G. (2015). Minimizing the number of early jobs on a proportionate flowshop. *Journal of the Operational Research Society, 66*(9), 1426–1429. https://doi.org/10.1057/jors.2014.112

Mor, B., & Mosheiov, G. (2016). Minsum and minmax scheduling on a proportionate flowshop with common flow-allowance. *European Journal of Operational Research, 254*(2), 360–370. https://doi.org/10.1016/j.ejor.2016.03.037

Mor, B., Mosheiov, G., & Shabtay, D. (2021). Minimizing the total tardiness and job rejection cost in a proportionate flow shop with generalized due dates. *Journal of Scheduling*. https://doi.org/10.1007/s10951-021-00697-4

Mor, B., & Shapira, D. (2019). Improved algorithms for scheduling on proportionate flowshop with job-rejection. *Journal of the Operational Research Society, 70*(11), 1997–2003. https://doi.org/10.1080/01605682.2018.1506540

Mor, B., & Shapira, D. (2020). Regular scheduling measures on proportionate flowshop with job rejection. *Computational and Applied Mathematics, 39*(2), 107. https://doi.org/10.1007/s40314-020-1130-z

Mosheiov, G., & Oron, D. (2012). Minimizing the number of tardy jobs on a proportionate flowshop with general position-dependent processing times. *Computers & Operations Research, 39*(7), 1601–1604. https://doi.org/10.1016/j.cor.2011.09.011

Ng, C. T., Cheng, T. C. E., & Yuan, J. J. (2006). A note on the complexity of the problem of two-agent scheduling on a single machine. *Journal of Combinatorial Optimization, 12*(4), 387–394. https://doi.org/10.1007/s10878-006-9001-0

Oron, D. (2019). Batching and resource allocation decisions on an m-machine proportionate flowshop. *Journal of the Operational Research Society, 70*(9), 1571–1578. https://doi.org/10.1080/01605682.2018.1495996

Ow, P. S. (1985). Focused Scheduling in Proportionate Flowshops. *Management Science, 31*(7), 852–869. https://doi.org/10.1287/mnsc.31.7.852

Panwalkar, S. S., & Koulamas, C. (2015). Proportionate flow shop: New complexity results and models with due date assignment. *Naval Research Logistics (NRL), 62*(2), 98–106. https://doi.org/10.1002/nav.21615

Panwalkar, S. S., & Koulamas, C. (2015). On equivalence between the proportionate flow shop and single-machine scheduling problems. *Naval Research Logistics (NRL), 62*(7), 595–603. https://doi.org/10.1002/nav.21666

Panwalkar, S. S., & Koulamas, C. (2017). On the dominance of permutation schedules for some ordered and proportionate flow shop problems. *Computers & Industrial Engineering, 107*, 105–108. https://doi.org/10.1016/j.cie.2017.03.013

Panwalkar, S. S., Smith, M. L., & Koulamas, C. (2013). Review of the ordered and proportionate flow shop scheduling research. *Naval Research Logistics (NRL), 60*(1), 46–55. https://doi.org/10.1002/nav.21518

Pourhejazy, P. (2024). Production management and supply chain integration. In J. Sarkis (Ed.), *The palgrave handbook of supply chain management.* Cham: Palgrave Macmillan. https://doi.org/10.1007/978-3-031-19884-7_86

Qian, J., & Han, H. (2022). Improved algorithms for proportionate flow shop scheduling with due-window assignment. *Ann Oper Res, 309*, 249–258. https://doi.org/10.1007/s10479-021-04414-4

Shabtay, D., & Oron, D. (2016). Proportionate flow-shop scheduling with rejection. *Journal of the Operational Research Society, 67*(5), 752–769. https://doi.org/10.1057/jors.2015.95

Shakhlevich, N., Hoogeveen, H., & Pinedo, M. (1998). Minimizing total weighted completion time in a proportionate flow shop. *Journal of Scheduling, 1*(3), 157–168. https://doi.org/10.1002/(SICI)1099-1425(1998100)1:3%3c157::AID-JOS12%3e3.0.CO;2-Y

Shiau, D., Cheng, S., & Huang, Y. (2008). Proportionate flexible flow shop scheduling via a hybrid constructive genetic algorithm. *Expert Systems with Applications, 34*(2), 1133–1143. https://doi.org/10.1016/j.eswa.2006.12.002

Sun, X., Geng, X.-N., & Liu, T. (2020). Due-window assignment scheduling in the proportionate flow shop setting. *Annals of Operations Research, 292*(1), 113–131. https://doi.org/10.1007/s10479-020-03653-1