Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Elsevier hereby grants permission to make all its COVID-19-related research that is available on the COVID-19 resource centre - including this research content - immediately available in PubMed Central and other publicly funded repositories, such as the WHO COVID database with rights for unrestricted research re-use and analyses in any form or by any means with acknowledgement of the original source. These permissions are granted for free by Elsevier for as long as the COVID-19 resource centre remains active.
No-wait two-stage flowshop problem with multi-task flexibility of the first machine

Jianming Dong, Hong Pan, Cunkui Ye, Weitian Tong, Jueliang Hu

PII: S0020-0255(20)30640-X
DOI: https://doi.org/10.1016/j.ins.2020.06.052
Reference: INS 15615

To appear in: Information Sciences

Received Date: 11 October 2019
Revised Date: 20 May 2020
Accepted Date: 25 June 2020

Please cite this article as: J. Dong, H. Pan, C. Ye, W. Tong, J. Hu, No-wait two-stage flowshop problem with multi-task flexibility of the first machine, Information Sciences (2020), doi: https://doi.org/10.1016/j.ins.2020.06.052

This is a PDF file of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability, but it is not yet the definitive version of record. This version will undergo additional copyediting, typesetting and review before it is published in its final form, but we are providing this version to give early visibility of the article. Please note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2020 Published by Elsevier Inc.
No-wait two-stage flowshop problem with multi-task flexibility of the first machine

Jianming Dong* Hong Pan* Cunkui Ye* Weitian Tong† Jueliang Hu*‡

May 19, 2020

Abstract

For the creation of intelligent management systems in hospitals, efficient resource arrangement is essential. Motivated by a real-world scenario in hospitals, we introduce the no-wait two-stage flowshop scheduling problem with the first-stage machine having multi-task flexibility. In this problem, each job has two operations which are processed in order on a two-stage flowshop without preemption and time delay between or on machines. The multi-task flexibility allows the first-stage machine to process the second-stage operations. The goal is to minimize the maximum completion time of all jobs. To the best of our knowledge, this is a pioneering work on this problem. We discover several novel structural properties, based on which we present a linear-time combinatorial algorithm with an approximation ratio $\frac{13}{8}$. This problem and its variants can find many other meaningful applications in modern manufacturing systems, such as the robot cell scheduling with computer numerical control machines or printed circuit boards. The idea behind our algorithm may inspire more practical algorithms.

Keywords: Flowshop; No-wait; Blocking; Multi-task flexibility; Worst-case performance ratio; Approximation algorithm

1 Introduction

Intelligent management systems in hospitals help to achieving a better and optimized use of intensive shared resources, which is a big challenge, particularly in the face of overwhelming demand and limited resources when a severe pandemic, such the Coronavirus [35], happens. Operating room and recovery room are two of the most resource-intensive areas in a hospital. For the purpose of reducing cost while maintaining a good quality of care, effective scheduling of operation room beds and recovery room beds has become one of the major priorities.

*Department of Mathematics, Zhejiang Sci-Tech University. Hangzhou, Zhejiang 310018, China.
†Department of Computer Science, Eastern Michigan University. Ypsilanti, MI 48165, USA.
‡Correspondence author. Email: hujlhz@163.com
Upon the completion of a surgery on the operating room bed, each patient should be transferred to a recovery bed as soon as possible to start the postanesthesia procedure. Operating room beds are capable of doing the operations of recovery room beds. In contrast, a recovery room bed is usually lack of necessary equipments for surgical operations. Thus unavailability of a bed in the recovery room at the end of a surgical operation can incur the beginning of postanesthesia procedure in the operating room until the discharge of a bed in the recovery room [14].

Motivated by the aforementioned scenario, we introduce and investigate a meaningful two-stage flowshop scheduling problem. Formally, there are two machines $M_1$ and $M_2$ and a job set $J = \{J_1, J_2, \ldots, J_n\}$. Each job $J_i, i \in \{1, 2, \ldots, n\}$ has two operations $A_i$ and $B_i$, named as the first- and second-stage operation respectively, which must be processed in order without preemption and any time delay. That is, 1) both $A_i$ and $B_i$ should be processed non-preemptively; 2) once $A_i$ is completed, $B_i$ has to be processed immediately; 3) $B_i$ cannot start processing until $A_i$ has been completely processed. Besides, we add flexibility for the first-stage machine $M_1$ such that it is able to process the second-stage operations, as an analogy with the relation between operating room and recovery room. We say $M_1$ has the **multi-task flexibility**. Our goal is to minimize the makespan, i.e. the maximum completion time of all jobs. To name our problem, we adopt the three-field notation $\alpha|\beta|\gamma$ introduced by Graham et al. [12]. In the $\alpha$ field (i.e. the scheduling environment), we use $F_2$ to represent the two-stage flowshop. In the $\beta$ field (i.e. the job characteristics and constraints), we use “nwt” to represent the no-wait constraint and “mflx $M_1 \rightarrow M_2$” to represent the multi-task flexibility of the first-stage machine. In the $\gamma$ field (i.e. the objective function), we use $C_{\text{max}}$ to represent the classic makespan minimization. Thus our problem can be denoted as $F_2|\text{nwt, mflx } M_1 \rightarrow M_2|C_{\text{max}}$.

Flowshop problems with either nwt or mflx constraint have been studied extensively in the literature [13, 4, 22, 18, 31]. However, the exploration of a flowshop under both the nwt and mflx constraints is rare. To the best of our knowledge, the $F_2|\text{nwt, mflx } M_1 \rightarrow M_2|C_{\text{max}}$ problem has never been studied before, though a preemptive version, i.e. $F_2|\text{nwt, mflx, prmp}|C_{\text{max}}$, was presented by Khorasanian and Moslehi [16], who gave two mathematical models optimally solving small-sized instances and a local search heuristic for the large-sized instances. We, on the other hand, focus on the design of approximation algorithm. Suppose $A$ is any approximation algorithm and $I$ is an instance. Let $C^A_{\text{max}}(I)$ and $C^*_{\text{max}}(I)$ denote the makespan generated by $A$ and the optimal algorithm, respectively. The performance ratio of the algorithm $A$ on $I$ is $\frac{C^A_{\text{max}}(I)}{C^*_{\text{max}}(I)}$. The algorithm $A$ is a $\rho$-approximation for a minimization problem if $\sup_I \frac{C^A_{\text{max}}(I)}{C^*_{\text{max}}(I)} \leq \rho$.

Our main contributes are summarized as follows.

- We propose a new scheduling model $F_2|\text{nwt, mflx } M_1 \rightarrow M_2|C_{\text{max}}$, which finds applications in hospital intelligent management systems. In addition, our $F_2|\text{nwt, mflx } M_1 \rightarrow M_2|C_{\text{max}}$ problem is able to model more real-world applications and thus has profound practical impact. The automated computer numerical control (CNC) machines are widely applied in metal cutting industries. They are highly flexible and can perform different operations as long as
the cutting tools required for these operations are loaded in the tool magazine of the machine [7]. Due to the expensive cost of the CNC machines, the multi-task flexibility is allowed on these machines to decrease the operating cost. Multi-task flexibility can also arise in the assembly of printed circuit boards (PCB), as the feeder tapes holding the components to be inserted may be present either on one machine only, or on several machines. In modern manufacturing systems, robots are installed in order to reduce labor cost, to increase output, to provide more flexible production systems and to replace people working in dangerous or hazardous conditions [5]. They are mainly used as material handling devices in robotic cells, where a robotic cell contains two or more robot-served machines [15]. There are no buffers at or between the machines, which requires the job scheduling to satisfy the no-wait constraints. Considering modern manufacturing systems where each robot cell contains CNC machines or PCBs, our $F_2|\text{nwt}, \text{mtflx} \ M_1 \rightarrow M_2|C_{\text{max}}$ problem can perfectly model these applications.

- We study the NP-hardness of the proposed problem and present a NP-hardness proof via a reduction from the PARTITION problem. We also design an approximation algorithm with a worst-case performance ratio of $\frac{13}{8}$. At a high level, we first obtain a non-trivial lower bound for the optimal solution; then propose two greedy algorithms, which utilize the no-wait and multi-task flexibility constraints to explore the structural properties of our problem; finally construct an intricate combinatorial algorithm by invoking the aforementioned two algorithms as subroutines. A case-by-case analysis shows the approximation ratio $\frac{13}{8}$. The discovered structural properties are of independent interest as it can be utilized to study other related scheduling problems.

In the following context, Section 2 introduces the most related works; In Section 3, we give necessary definitions and notations, present a lower bound for the optimal makespan, and show a trivial NP-hardness proof for the $F_2|\text{nwt}, \text{mtflx} \ M_1 \rightarrow M_2|C_{\text{max}}$ problem. Our $\frac{13}{8}$-approximation algorithm is presented in Section 4 and analyzed in Section 5. Section 6 concludes the paper and proposes several directions for the future work.

2 Related works

There is an extremely rich literature on the flowshop scheduling problems with $\text{nwt}$ (i.e. no-wait) or $\text{mtflx}$ (i.e. multi-task flexibility) constraint. In this section, we review most related works mainly along these two directions.

2.1 Flowshop scheduling with no-wait constraint

In the traditional flowshop scheduling problem, all jobs need to be processed in order on the flowshop machines, i.e. each job starts on the first-stage machine, then it is processed on the second-stage
machine, up to the last machine. The intermediate storage capacity between machines, named as \textit{buffer}, are considered infinite and machines are always available for processing jobs [20].

When buffers between machines are not available, i.e. intermediate storage capacity is considered zero, blocking occurs, as a job, having completed processing on a machine, remains on the machine until the next-stage machine becomes available for processing. Such a scenario is characterized as the \textit{blocking} environment. The \textit{no-wait} environment is similar to the blocking environment but more restrictive by requiring that a job must be processed from start to completion without any interruption either on or between machines. Under the three-field notation, the no-wait or blocking flow-shop scheduling problem with makespan minimization is denoted as $F_m|\textit{nwt}|C_{\text{max}}$ or $F_m|\textit{blocking}|C_{\text{max}}$, respectively.

Flowshop scheduling under the no-wait or blocking environment has been investigated extensively in the literature. In particular, Reddi and Ramamoorthy [29] proved $F_2|\textit{nwt}|C_{\text{max}}$ is equivalent to $F_2|\textit{blocking}|C_{\text{max}}$. It was shown that $F_2|\textit{nwt}|C_{\text{max}}$ can be reduced to a special case of the Traveling Salesman problem [26, 29] and can be solved in polynomial time by the Gilmore-Gomory algorithm [10]. When $m \geq 3$, $F_m|\textit{nwt}|C_{\text{max}}$ is strongly NP-hard [30, 21]. Because of the complex nature of the general $F_m|\textit{nwt}|C_{\text{max}}$ problem, exact methods are used to solve small instances while heuristics and metaheuristics methods are more common for larger instances. Examples can be found in [28, 23, 1, 24, 2]. As the $F_2|\textit{nwt}|C_{\text{max}}$ problem is polynomially solvable, the following works either provide computationally efficient heuristics or consider its variant by changing environment constraints and/or adopting different objective functions. Glass, Gupta, and Potts [11] considered the case where jobs might have missing operations in $F_2|\textit{nwt}|C_{\text{max}}$ and presented an efficient heuristic with a worst-case ratio of 4/3. Wang, Yang, and Lin [34] proposed simulated annealing and genetic algorithms for $F_2|\textit{nwt}|C_{\text{max}}$. Espinouse, Formanowicz, and Penz [9, 8] studied the complexity of the $F_2|\textit{nwt}|C_{\text{max}}$ problem with several machine availability constraints and proposed heuristic algorithms. Wang et al. [32] investigated a hybrid variant of $F_2|\textit{nwt}|C_{\text{max}}$, where the first stage contains a single machine and the second stage contains several identical parallel machines, and designed a branch-and-bound algorithm. Allahverdi et al. [3] established local and global dominance relations to solve the no-wait two-stage flowshop scheduling problem to minimize maximum lateness, where setup times are considered separate from processing times. There are plenty other works on the no-wait or blocking flowshop scheduling. Readers may refer to three excellent surveys [13, 4, 22]. More specifically, Hall and Sriskandarajah [13] presented an excellent review of the literature, covering about 130 papers, on scheduling problems (including flowshop, job shop, and open shop) with no-wait and/or blocking in process since 1970s until mid-1993; The continuing survey paper by Allahverdi [4] provides analysis and an extensive review of more than 300 papers that appeared since the mid-1993 to the beginning of 2016; Miyata and Nagano [22] presented a thorough review on the flowshop scheduling problem with blocking conditions, covering 139 papers ranging from 1969 up to early 2019.
2.2 Flowshop scheduling with multi-task flexibility

The multi-task flexibility allows the machine at one stage to process operations at the other stages. Suppose \( M_i \rightarrow M_k \) denotes that the machine at the \( i \)-th stage is capable of processing operations at the \( k \)-th stage. Let \( M_i \leftrightarrow M_k \) indicate that \( M_i \rightarrow M_k \) and \( M_k \rightarrow M_i \) holds simultaneously.

Lee and Mirchandani \[17\] introduced and investigated the \( F_2|mtflx M_1 \leftrightarrow M_2|C_{\text{max}} \) problem. They studied its NP-hardness and presented an effective heuristic algorithm. Liao et al. \[19\] considered the \( F_m|mtflx|C_{\text{max}} \) problem and presented two mixed integer programming models, one is for the case where the job sequence is given and the other is for the case where the job sequence is to be determined. Pan and Chen \[25\] studied three possible cases of the \( F_2|mtflx|C_{\text{max}} \) problem, i.e. \( F_2|mtflx M_1 \leftrightarrow M_2|C_{\text{max}} \), \( F_2|mtflx M_1 \rightarrow M_2|C_{\text{max}} \), \( F_2|mtflx M_1 \leftarrow M_2|C_{\text{max}} \). The NP-hardness was proved for all cases. In addition, they developed three branch and bound algorithms. Cheng and Wang \[6\] proved the NP-hardness for the \( F_2|mtflx M_1 \leftrightarrow M_2|C_{\text{max}} \) problem in the ordinary sense and presented a pseudo-polynomial dynamic programming approach. Motivated by the applications in image processing, Wei and He \[27\] studied a general variant of the \( F_2|mtflx M_1 \leftarrow M_2|C_{\text{max}} \) problem, where each first-stage operation \( A_i \) needs more processing time on \( M_2 \) compared with \( M_1 \). They provided a pseudo-polynomial time optimal algorithm and a polynomial time approximation algorithm with a tight worst-case ratio 2. Wei et al. \[33\] gave a fully polynomial time approximation scheme for the \( F_2|mtflx M_1 \leftarrow M_2|C_{\text{max}} \) problem.

3 Preliminaries

Recall that in the \( F_2|nwt, mtflx M_1 \rightarrow M_2|C_{\text{max}} \) problem, each job \( J_i, i \in \{1, 2, \ldots, n\} \) needs to be processed non-preemptively without any time delay and the machine \( M_1 \) can process the second-stage operations. Suppose \( A_i \) and \( B_i \) have the processing time \( a_i \) and \( b_i \) respectively. Let \( p_i = a_i + b_i \) denote the total processing time of the job \( J_i \). We remark that \( B_i \) cannot start processing until \( A_i \) has been completed and \( B_i \) has the same processing time no matter on \( M_1 \) or \( M_2 \). Let \([n]\) denote the integer set \( \{1, 2, \ldots, n\} \) for any positive integer \( n \). As all indexes in the following context are integers, \( \ell \in [i, j] \) represents \( \ell \in \{i, i+1, \ldots, j\} \) for an index \( \ell \). Let \( a = \sum_{i \in [n]} a_i \) and \( b = \sum_{i \in [n]} b_i \) denote the total processing time of operations on \( M_1 \) and \( M_2 \) respectively. For a feasible schedule \( \pi \), its makespan is defined as \( C_{\text{max}}^\pi \). Suppose \( \pi^* \) is the optimal schedule and \( C_{\text{max}}^{\pi^*} \) is its makespan. We use \( C_{\text{max}}^* \) to represent \( C_{\text{max}}^{\pi^*} \) for the sake of simplicity.

**Theorem 3.1** The optimal makespan \( C_{\text{max}}^* \) for the \( F_2|nwt, mtflx M_1 \rightarrow M_2|C_{\text{max}} \) problem can be lower bounded by

\[
LB = \lambda_1 \cdot \max_{i \in [n]} p_i + \lambda_2 \cdot \left( a + \min_{i \in [n]} b_i \right) + \lambda_3 \cdot \left( b + \min_{i \in [n]} a_i \right)
\]

with \( \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0, \lambda_1 + \lambda_2 + \lambda_3 = 1 \).
Proof. We only need to prove the lower bound
\[
\max \left\{ \max_{i \in [n]} \{ p_i \}, a + \min_{i \in [n]} \{ b_i \}, b + \min_{i \in [n]} \{ a_i \} \right\}.
\]

For any job, its two operations needs to be processed in order and without any interruption either on or between machines. Therefore, the first term is trivial. When \( M_1 \) finishes processing all first-stage operations, at least one second-stage operation needs to be processed, from which the second term follows. As the second-stage operation of each job can only be processed after the corresponding first-stage operation is finished on \( M_1 \), there is a necessary idle time slot of a duration at least \( \min_{i \in [n]} \{ a_i \} \) on \( M_2 \) before it starts processing any operation. Thus, the third term follows.

\[ \square \]

**Theorem 3.2** The \( F_2|\text{nwt, mflx} M_1 \rightarrow M_2|C_{\max} \) problem is NP-hard.

The proof is similar to the NP-hardness proof for the \( F_2|\text{mflx} M_1 \rightarrow M_2|C_{\max} \) problem [25].

Proof. The main idea is constructing a reduction from the PARTITION problem, which decides whether a given multiset \( S \) of \( n \) positive integers \( \{ s_1, s_2, \ldots, s_n \} \) can be partitioned into two subsets \( S_1 \) and \( S_2 \) such that the sum of the numbers in \( S_1 \) equals the sum of the numbers in \( S_2 \).

Given a PARTITION instance, we define a \( F_2|\text{nwt, mflx} M_1 \rightarrow M_2|C_{\max} \) instance with each job \( J_i \) having processing time \( a_i = 0, b_i = s_i, i \in [n] \). According to the lower bound for the optimal makespan we obtained in Theorem 3.1, the optimal makespan of the constructed instance is at least \( \frac{1}{2} \sum_{i \in [n]} s_i \). We can easily observe that the PARTITION instance has a Yes answer if and only if the constructed instance has a makespan of \( \frac{1}{2} \sum_{i \in [n]} s_i \).

\[ \square \]

4 Algorithm Description

In this section, we give descriptions for two versions of the \( \frac{13}{8} \)-approximation algorithm.

We sort jobs in non-increasing order with respect to the size of the second-stage operation. In sequel, we assume the job sequence \( \langle J_1, J_2, \ldots, J_n \rangle \) satisfies \( b_1 \geq b_2 \geq \ldots \geq b_n \) without loss of generality. Next, we introduce the key concepts, critical job and critical position. The critical position is the very first position in the job sequence such that \( b_1 \) is no larger than the total processing time of the second-stage operations for jobs before this position (inclusively). The critical job is the job at the critical position.

**Definition 1 (Critical job)** A job \( J_k, k \in [n] \) is critical in \( \langle J_1, J_2, \ldots, J_n \rangle \) if \( \sum_{\ell \in [2, k-1]} b_\ell < b_1 \leq \sum_{\ell \in [k, n]} b_\ell \). And we say \( k \) is the critical position.
We say a machine is *idle* or *not busy* if it is not processing any operations. A flowshop is *not busy* before it completes processing all jobs if both machines of the flowshop have the overlapping idle time slot before the makespan. If the flowshop is not busy under a feasible schedule, we can always request the flowshop to process the unfinished operations earlier to fill this idle gap and thus obtain a feasible schedule with a smaller makespan. Without loss of generality, in the following discussions, we only consider the feasible schedules that keep the flowshop always busy.

Roughly, our algorithm finds the critical job and then schedule the jobs before and after the critical position by invoking two greedy subroutines. Both subroutines take advantage of the multi-task flexibility of the first-stage machine and schedule a part of the second-stage operations on $M_1$ such that the flowshop is always busy and the overlapping processing time between $M_1$ and $M_2$, i.e. the total amount of time that $M_1$ and $M_2$ are simultaneously processing operations, is greedily maximized. This is why our algorithm is named as MAX-OVERLAP. Let $o$ denote the overlapping processing time between $M_1$ and $M_2$.

**Lemma 4.1** For any feasible schedule $\pi$, the makespan can be computed as

$$C_{\text{max}}^\pi = a + b - o.$$ 

**Proof.** Suppose $T_1$ is the total time before the makespan that $M_1$ is busy while $M_2$ is idle. Similarly, we can define $T_2$. As $o$ is the overlapping processing time between $M_1$ and $M_2$, the total processing time of operations on the flowshop is $a + b = T_1 + o + T_2 + o$. Therefore, the makespan is $C_{\text{max}}^\pi = T_1 + o + T_2 = a + b - o$. This proves the lemma. 

The first subroutine, denoted as $\text{Alg1}(g,i,j)$, takes in a subsequence of jobs $\langle J_i, J_{i+1}, \ldots, J_j \rangle$, $i \leq j \in [n]$ and a job $J_g, g \not\in [i,j]$. ALG1 considers the case where $J_g$ has a relatively large second-stage operation with respect to the total processing time of $\langle J_i, J_{i+1}, \ldots, J_j \rangle$. When we call $\text{Alg1}(g,i,j)$, let $M_1$ start processing $A_g$ as early as possible following the existing subschedule on the flowshop and greedily schedule the jobs $\langle J_i, J_{i+1}, \ldots, J_j \rangle$ in order on $M_1$ right after $A_g$ without any delay. Refer to Algorithm 1 for detailed description and Figure 1 for a visualized demo. As a result, the flowshop is always busy when processing the job set $\{J_g\} \cup \{J_i, J_{i+1}, \ldots, J_j\}$. Then, we have the following lemma via a simple discussion whether $b_g \leq \sum_{\ell \in [i,j]} p_\ell$ holds.

**Lemma 4.2** During the time interval that the flowshop processes the job set $\{J_g\} \cup \{J_i, J_{i+1}, \ldots, J_j\}$, the overlapping time between $M_1$ and $M_2$ can be quantitatively measured by

$$o = \min \left\{ b_g, \sum_{\ell \in [i,j]} p_\ell \right\}.$$
Algorithm 1 \textbf{ALG1}(g, i, j)

\textbf{Input:} a subsequence of jobs \( \langle J_i, J_{i+1}, \ldots, J_j \rangle \), \( i \leq j \in [n] \), a job \( J_g, g \notin [i, j] \);
\textbf{Output:} a feasible schedule for \( \{ J_g \} \cup \{ J_i, J_{i+1}, \ldots, J_j \} \) such that the flowshop is always busy and the overlapping time is large.

1: Let \( \sigma_1 = \langle A_g, A_i, B_i, A_{i+1}, B_{i+1}, \ldots, A_j, B_j \rangle \); \hfill \# Sub-schedule on \( M_1 \)
2: Start processing \( \sigma_1 \) as early as possible;
3: Let \( \sigma_2 = \langle B_g \rangle \); \hfill \# Sub-schedule on \( M_2 \)
4: Start processing \( \sigma_2 \) as early as possible;
5: \textbf{return} \( \pi = (\sigma_1, \sigma_2) \);

The second subroutine, denoted as \textbf{ALG2}(i, j), takes in a subsequence of jobs \( \langle J_i, J_{i+1}, \ldots, J_j \rangle \), \( i \leq j \in [n] \). We pair up the adjacent jobs \( \{ J_\ell, J_{\ell+1} \}, \ell \in \{ i, i+2, i+4, \ldots \} \). Then we swap the jobs in the same pair and obtain a new job sequence \( \langle J_{i+1}, J_i, J_{i+3}, J_{i+2}, \ldots \rangle \). If there are even number of jobs, the reordered job sequence is \( \langle J_{i+1}, J_i, J_{i+3}, J_{i+2}, \ldots, J_j \rangle \); otherwise, the reordered job sequence is \( \langle J_{i+1}, J_i, J_{i+3}, J_{i+2}, \ldots, J_{j-1}, J_{j-2}, J_j \rangle \). We alternatively schedule the jobs such that one is processed on both machines and the other one is completely processed on \( M_1 \). Refer to Algorithm 2 for detailed description and Figure 2 for a visualized demo. As \( b_i \geq b_{i+1} \geq \ldots \geq b_j \), we have \( p_\ell = a_\ell + b_\ell \geq b_{\ell+1} \) for \( \ell \in [i, j-1] \). Therefore, the schedule returned by \textbf{ALG2}(i, j) makes the flowshop always busy. We estimate the overlapping time between \( M_1 \) and \( M_2 \) in Lemma 4.3.

**Lemma 4.3** During the time interval that the flowshop processes the job set \( \{ J_i, J_{i+1}, \ldots, J_j \} \), the overlapping time between \( M_1 \) and \( M_2 \) can be lower bounded as follows.

\[ o \geq \frac{1}{2} \sum_{\ell \in [i+1,j]} b_\ell \geq \frac{b}{2} - \frac{1}{2} \sum_{\ell \in [i]} b_\ell. \]

**Proof.**

If there are even number of jobs, the reordered job sequence is \( \langle J_{i+1}, J_i, J_{i+3}, J_{i+2}, \ldots, J_j \rangle \). For each \( \ell \in \{ i, i+2, i+4, \ldots \} \), \( J_\ell \) is completely processed on \( M_1 \) while \( J_{\ell+1} \) is processed on both machines. Because of \( p_\ell = a_\ell + b_\ell \geq b_{\ell+1} \), when \( M_1 \) starts processing \( J_\ell \), \( M_2 \) starts processing \( B_{\ell+1} \).
(Refer to Figure 2) The overlapping time can be calculated as

\[ o = b_{i+1} + b_{i+3} + \ldots + b_{j-2} + b_j \]
\[ \geq \frac{b_{i+1} + b_{i+2}}{2} + \frac{b_{i+3} + b_{i+4}}{2} + \ldots + \frac{b_{j-2} + b_{j-1}}{2} + b_j \]
\[ \geq \frac{1}{2} \sum_{\ell \in [i+1, j]} b_\ell, \]

where the first inequality is because of \( b_i \geq b_{i+1} \geq \ldots \geq b_j \).

When there are odd number of jobs, the overlapping time can be computed similarly.

\[ o = b_{i+1} + b_{i+3} + \ldots + b_{j-3} + b_{j-1} \]
\[ \geq \frac{b_{i+1} + b_{i+2}}{2} + \frac{b_{i+3} + b_{i+4}}{2} + \ldots + \frac{b_{j-2} + b_{j-1}}{2} + \frac{b_{j-1} + b_j}{2} \]
\[ = \frac{1}{2} \sum_{\ell \in [i+1, j]} b_\ell. \]

This proves the lemma.

\[ \square \]

Figure 2: An illustration for \text{Alg2}(i, j)

| M_1 | A_{i+1} | J_i | A_{i+3} | J_{i+2} | A_{i+5} | J_{i+4} | \ldots |
|-----|--------|-----|--------|--------|--------|--------|------|
| M_2 | B_{i+1} | B_{i+3} | B_{i+5} | \ldots |

\[ t \]

\textbf{Algorithm 2} \text{Alg2}(i, j)

\textbf{Input:} a subsequence of jobs \( \langle J_i, J_{i+1}, \ldots, J_j \rangle \), \( i \leq j \in [n] \);

\textbf{Output:} a feasible schedule for \( \{ J_i, J_{i+1}, \ldots, J_j \} \) such that the flowshop is always busy and the overlapping time is large.

1: Let \( \sigma_1 = \langle A_{i+1}, A_i, B_i, A_{i+2}, A_{i+3}, B_{i+3}, \ldots \rangle \); \hspace{1cm} \# Sub-schedule on \( M_1 \)
2: Start processing \( \sigma_1 \) as early as possible;
3: Let \( \sigma_2 = \langle B_{i+1}, B_{i+3}, \ldots \rangle \); \hspace{1cm} \# Sub-schedule on \( M_2 \)
4: Start processing \( \sigma_2 \) as early as possible;
5: return \( \pi = (\sigma_1, \sigma_2) \);

Recall that \( b_1 \) has relatively large with respect to the total processing time of the second-stage operation of jobs before the critical position if it exists. More specifically, \( \sum_{\ell \in [2, k-1]} b_\ell < b_1 \leq \sum_{\ell \in [k, n]} b_\ell \) for the critical position \( k \). The rough idea of the \text{Max-Overlap} algorithm is to schedule the jobs sequences \( \langle J_1, \ldots, J_{k-1} \rangle \) and \( \langle J_k, \ldots, J_n \rangle \) with \text{Alg1} and \text{Alg2}, respectively. However,
there are two problems: 1) the critical position may not exist; 2) when the critical position is close to the boundary, i.e. 1 or n, the trivial combination of ALG1 and ALG2 is not able to achieve the approximation ratio $13/8$. Therefore, our algorithm MAX-OVERLAP needs to handle these special cases carefully. Refer to Algorithm 3 for a detailed description of the MAX-OVERLAP algorithm.

5 The worst-case performance ratio analysis

In this section, we will prove that the worst-case performance ratio of the MAX-OVERLAP algorithm is $13/8$. As shown in Algorithm 3, there are two versions for the MAX-OVERLAP algorithm. The first version is the brute-force implementation of the second version. It simply computes 9 feasible schedule candidates and outputs the one with the minimum makespan. This version is easy to implement but provides little information for the worst-case analysis. On the other hand, the second version offers a detailed case-by-case discussion to avoid the unnecessary computation of some schedule candidates in the first version. Such discussions are helpful to our analysis. We summarize the cases and corresponding analyses in Table 1.

| Cases | size of $b_1$ | critical position $k$ | Candidate schedules | Analysis |
|-------|---------------|------------------------|---------------------|----------|
| Case 1 | $b_1 \geq 3/8 LB$ | $k$ | $\pi_1$ | Lemma 5.1 |
| Case 2 | $b_1 \leq 3/8 LB$ | $k$ | $\pi_2$ | Lemma 5.2 |
| Case 3 | $2/8 LB < b_1 < 3/8 LB$ | not available | $\pi_1$ | Lemma 5.3 |
| Case 4 | $2/8 LB < b_1 < 3/8 LB$ | $k \geq 4$ | $\pi_1, \pi_3$ | Lemma 5.4 |
| Case 5 | $2/8 LB < b_1 \leq 5/16 LB$ | $k \leq 3$ | $\pi_4, \pi_5, \pi_6$ | Lemma 5.5 |
| Case 6 | $5/16 LB < b_1 < 3/8 LB$ | $k \leq 3$ | $\pi_4, \pi_5, \pi_6, \pi_7, \pi_8, \pi_9$ | Lemma 5.6 |

Lemma 5.1 In Case 1, i.e. $b_1 \geq 3/8 LB$, we have $C_{\max}^{\pi_1} \leq 13/8 C_{\max}$.

PROOF. In this case, we consider the schedule $\pi_1$, which is returned by ALG1(1, 2, n). By Lemma 4.2 and Lemma 5.1

$$C_{\max}^{\pi_1} = a + b - o = a + b - \min \left\{ b_1, \sum_{\ell \in [2,n]} p_\ell \right\}.$$  

By Theorem 3.1 we have $C_{\max}^* \geq LB \geq \max \{ a + b/2, p_1 \}$. When $b_1 \leq \sum_{\ell \in [2,n]} p_\ell$, we have

$$C_{\max}^{\pi_1} = a + b - b_1 \leq 2LB - 3/8 LB = 13/8 LB \leq 13/8 C_{\max}^*,$$

where the first inequality follows from $b_1 \geq 3/8 LB$. When $b_1 > \sum_{\ell \in [2,n]} p_\ell$, we have

$$C_{\max}^{\pi_1} = a + b - \sum_{\ell \in [2,n]} p_\ell = a_1 + b_1 = p_1 \leq C_{\max}^*.$$
Algorithm 3  Max-Overlap

**Input:** a job sequence \((J_1, J_2, \ldots, J_n)\) with \(b_1 \geq b_2 \geq \ldots \geq b_n\); 
**Output:** a feasible schedule such that the flowshop is always busy and the overlapping time is large.

▷ A brute-force version:

1: \(\pi_1 = \text{ALG1}(1, 2, n)\);
2: \(\pi_2 = \text{ALG2}(1, n)\);
3: \(\pi_3 = \text{ALG1}(1, 2, k - 1)\) followed by \(\text{ALG2}(k, n)\);
4: \(\pi_4 = \text{ALG1}(1, 2, 3)\) followed by \(\text{ALG2}(4, n)\);
5: \(\pi_5 = \text{ALG1}(1, 2, 3)\) followed by \(\text{ALG1}(4, 5, n)\);
6: \(\pi_6 = \text{ALG1}(1, 2, 2)\) followed by \(\text{ALG1}(3, 4, n)\);
7: \(\pi_7 = \text{ALG1}(3, 4, n)\) followed by \(\text{ALG1}(1, 2, 2)\);
8: \(\pi_8 = \text{ALG1}(1, 4, n)\) followed by \(\text{ALG1}(2, 3, 3)\);
9: \(\pi_9 = \text{ALG1}(1, 4, n)\) followed by \(\text{ALG1}(3, 2, 2)\);
10: \(\pi = \arg\min_{\pi \in \{\pi_1, \pi_2, \ldots, \pi_9\}} \{C_{\pi}^{\max}\}\);
11: return \(\pi\);

▷ An efficient case-by-case version:

12: if \(b_1 \geq \frac{3}{8} LB\) then 
13: \(\pi = \pi_1\); \# Case 1
14: else if \(b_1 \leq \frac{2}{3} LB\) then 
15: \(\pi = \pi_2\); \# Case 2
16: else 
17: if critical position \(k\) does not exist then 
18: \(\pi = \pi_1\); \# Case 3
19: else 
20: if \(k \geq 4\) then 
21: \(\pi = \arg\min_{\pi \in \{\pi_3, \pi_4\}} \{C_{\pi}^{\max}\}\); \# Case 4
22: else 
23: if \(b_1 > \frac{2}{5} LB\) and \(b_1 < \frac{5}{10} LB\) then 
24: \(\pi = \arg\min_{\pi \in \{\pi_5, \pi_6\}} \{C_{\pi}^{\max}\}\); \# Case 5
25: else 
26: \(\pi = \arg\min_{\pi \in \{\pi_7, \pi_8, \pi_9\}} \{C_{\pi}^{\max}\}\); \# Case 6
27: end if 
28: end if 
29: end if 
30: end if 
31: return \(\pi\);
This proves the lemma.

**Lemma 5.2** In Case 2, i.e. \( b_1 \leq \frac{2}{8}LB \), we have \( C_{\max}^{\pi_2} \leq \frac{13}{8}C^{*}_{\max} \).

**Proof.** In this case, we consider the schedule \( \pi_2 \), which is returned by \( \text{Alg2}(1,n) \). By Lemma 4.3 and Lemma 4.1,

\[
C_{\max}^{\pi_2} = a + b - o = a + b - \left( \frac{b}{2} - \frac{1}{2}b_1 \right) = a + \frac{b}{2} + \frac{1}{2}b_1.
\]

By Theorem 3.1, we have \( C^{*}_{\max} \geq LB \geq \max \{ \frac{a+b}{2}, a, p_1 \} \). Then we have

\[
C_{\max}^{\pi_2} = a + \frac{b}{2} + \frac{1}{2}b_1 \leq \frac{1}{2}LB + \frac{1}{2} \cdot 2LB + \frac{1}{2} \cdot \frac{2}{8}LB = \frac{13}{8}LB \leq \frac{13}{8}C^{*}_{\max},
\]

where the first inequality follows from \( b_1 \leq \frac{2}{8}LB \). This proves the lemma.

**Lemma 5.3** In Case 3, i.e. \( \frac{2}{8}LB < b_1 < \frac{3}{8}LB \) and the critical position does not exist, we have \( C_{\max}^{\pi_1} \leq \frac{11}{8}C^{*}_{\max} \).

**Proof.** In this case, we consider the schedule \( \pi_1 \), which is returned by \( \text{Alg1}(1,2,n) \). As the critical position does not exist, it must be the case

\[
b_1 > \sum_{\ell \in [2,n]} b_{\ell} = b - b_1. \tag{5.1}
\]

By Lemma 4.2 and Lemma 4.1,

\[
C_{\max}^{\pi_1} = a + b - o = a + b - \min \left\{ b_1, \sum_{\ell \in [2,n]} p_{\ell} \right\} = \max \{ a + b - b_1, a_1 + b_1 \} \leq a + b_1 \leq LB + \frac{3}{8}LB = \frac{11}{8}LB \leq \frac{11}{8}C^{*}_{\max},
\]

where the first inequality follows from Eq. (5.1); the third inequality is because of \( C^{*}_{\max} \geq LB \geq a \) and \( b_1 < \frac{3}{8}LB \). This proves the lemma.

**Lemma 5.4** In Case 4, i.e. \( \frac{2}{8}LB < b_1 < \frac{3}{8}LB \) and the critical position \( k \geq 4 \), we have \( \min \{ C_{\max}^{\pi_1}, C_{\max}^{\pi_3} \} \leq \frac{13}{8}C^{*}_{\max} \).

**Proof.** As \( b_1 \geq b_2 \geq \ldots \geq b_n \) and \( k \geq 4 \), we have

\[
\sum_{\ell \in [2,k-2]} b_{\ell} \geq \frac{1}{3} \sum_{\ell \in [2,k]} b_{\ell} \geq \frac{1}{3}b_1, \tag{5.2}
\]
where the last inequality holds due to the definition of the critical job.

**Case 4.1** \( k < n \): we consider the schedule \( \pi_3 \), which is the sub-schedule \( \text{Alg1}(1, 2, k - 1) \) followed by the sub-schedule \( \text{Alg2}(k, n) \). By Lemma 4.2 and Lemma 4.3, the total overlapping time between \( M_1 \) and \( M_2 \) can be lower bounded as

\[
o \geq \min \left\{ b_1, \sum_{\ell \in [2,k-1]} p_\ell \right\} + \frac{b}{2} - \frac{1}{2} \sum_{\ell \in [k]} b_\ell
\]

\[
\geq \sum_{\ell \in [2,k-1]} b_\ell + \frac{b}{2} - \frac{1}{2} \sum_{\ell \in [k]} b_\ell = \frac{b}{2} - \frac{b_1}{2} + \frac{1}{2} \sum_{\ell \in [2,k-1]} b_\ell - \frac{1}{2} b_k
\]

\[
\geq \frac{b}{2} - \frac{b_1}{2} + \frac{1}{2} \sum_{\ell \in [2,k-2]} b_\ell \geq \frac{b}{2} - \frac{b_1}{3}.
\]

where the second inequality follows from the definition of the critical job; the third inequality is due to the fact \( b_{k-1} \geq b_k \); the last inequality is because of Eq.(5.2).

By Theorem 3.1, we have \( C^*_\text{max} \geq LB \geq \max\{\frac{a+b}{2}, a\} \). Combining Lemma 4.1 and \( b_1 < \frac{3}{8} LB \), we have

\[
C^*_{\pi_3} = a + b - o \leq a + b - \left( \frac{b}{2} - \frac{b_1}{3} \right) = \frac{1}{2} a + \frac{1}{2} (a + b) + \frac{1}{3} b_1
\]

\[
\leq \frac{1}{2} LB + \frac{1}{2} \cdot 2LB + \frac{1}{3} \cdot \frac{3}{8} LB = \frac{13}{8} LB \leq \frac{13}{8} C^*_\text{max}
\]

**Case 4.2** \( k = n \): we consider the schedule \( \pi_1 \), which is returned by \( \text{Alg1}(1, 2, n) \). According to the definition of the critical job, \( b_1 > \sum_{\ell \in [2,n-1]} b_\ell \). As \( k \geq 4 \) holds and \( b_1 \geq b_2 \geq \ldots \geq b_n \), we have \( b_1 > 2b_n \), from which we have

\[
\sum_{\ell \in [2,n]} b_\ell = \sum_{\ell \in [2,n-1]} b_\ell + b_n \leq \frac{3}{2} b_1.
\]

(5.3)

From Lemma 4.2, Lemma 4.1, and \( C^*_\text{max} \geq LB \geq a \), the makespan can be computed as

\[
C^*_{\pi_1} = a + b - o = a + b - \min \left\{ b_1, \sum_{\ell \in [2,n]} p_\ell \right\} = \max\{a + b - b_1, a_1 + b_1\}
\]

\[
\leq a + \frac{3}{2} b_1 \leq LB + \frac{3}{2} \cdot \frac{3}{8} LB = \frac{25}{16} LB < \frac{13}{8} LB \leq \frac{13}{8} C^*_\text{max},
\]

where the first inequality is because of Eq.(5.3); the second inequality is due to the fact \( b_1 < \frac{3}{8} LB \).

To summarize, \( \min\left\{ C^*_{\pi_1}, C^*_{\pi_3} \right\} \leq \frac{13}{8} C^*_\text{max} \) holds in Case 4. This proves the lemma.

The following lemmas discuss the cases when the critical position \( k \leq 3 \). To avoid trivial cases, we assume there are at least five jobs, i.e. \( n \geq 5 \).

**Lemma 5.5** In Case 5, i.e. \( \frac{3}{8} LB < b_1 < \frac{5}{16} LB \) and the critical position \( k \leq 3 \), we have \( \min\{C^*_{\pi_4}, C^*_{\pi_5}, C^*_{\pi_6}\} \leq \frac{13}{8} C^*_\text{max} \).
PROOF. By the definition of the critical job and the fact that \( b_1 \geq b_2 \geq \ldots \geq b_n \), we have \( b_1 \leq b_2 + b_3 \leq 2b_1 \). Depending on the value of \( b_2 + b_3 \), we discuss two cases, \( b_2 + b_3 \leq \frac{3}{2}b_1 \) and \( b_2 + b_3 > \frac{3}{2}b_1 \).

**Case 5.1** \( b_2 + b_3 \leq \frac{3}{2}b_1 \): we investigate three subcases based on the value of \( b_4 \).

- **Case 5.1.1** \( b_4 \leq \frac{1}{3} \sum_{\ell \in [4,n]} b_\ell \): In this subcase, we consider the schedule \( \pi_4 \), which is ALG1(1,2,3) followed by ALG2(4, n).

As \( b_4 \leq \frac{1}{3} \sum_{\ell \in [4,n]} b_\ell \) holds in this subcase, we have \( 2b_4 \leq \sum_{\ell \in [5,n]} b_\ell \) and thus

\[
3 \cdot \sum_{\ell \in [5,n]} b_\ell \geq 2 \sum_{\ell \in [5,n]} b_\ell + 2b_4 = 2 \sum_{\ell \in [4,n]} b_\ell.
\]  

(5.4)

By Lemma 4.2 and Lemma 4.3, the total overlapping time between \( M_1 \) and \( M_2 \) is

\[
o \geq \min \left\{ b_1, \sum_{\ell \in [2,3]} p_\ell \right\} + \frac{1}{2} \sum_{\ell \in [5,n]} b_\ell \\
\geq b_1 + \frac{1}{2} \sum_{\ell \in [5,n]} b_\ell \geq b_1 + \frac{1}{3} \sum_{\ell \in [4,n]} b_\ell \\
= b_1 + \frac{1}{3} \left( b - \sum_{\ell \in [3]} b_\ell \right) \geq b_1 + \frac{1}{3} \left( b - \frac{5}{2} b_1 \right) = \frac{1}{3} b + \frac{1}{6} b_1,
\]

where the second inequality follows from \( b_1 \leq b_2 + b_3 \); the third inequality is because of Eq. (5.4); the last inequality is due to \( b_2 + b_3 \leq \frac{3}{2} b_1 \). By Theorem 3.1, we have \( C_{\max}^* \geq \frac{13}{8} \), and thus

\[
C_{\max}^{\pi_4} = a + b - o = a + b - \left( \frac{1}{3} b + \frac{1}{6} b_1 \right) = \frac{1}{3} a + \frac{2}{3} (a + b) - \frac{1}{6} b_1 \\
\leq \frac{1}{3} \frac{13}{8} LB + \frac{2}{3} \cdot 2LB - \frac{1}{6} \cdot \frac{2}{8} LB = \frac{13}{8} \frac{13}{8} LB \leq \frac{13}{8} C_{\max}^*.
\]

**Case 5.1.2** \( b_4 \in \left( \frac{2}{3} \sum_{\ell \in [4,n]} b_\ell, \frac{2}{3} \sum_{\ell \in [4,n]} b_\ell \right) \): In this subcase, we consider the schedule \( \pi_5 \), which is ALG1(1,2,3) followed by ALG1(4, 5, n).

As \( b_4 < \frac{2}{3} \sum_{\ell \in [4,n]} b_\ell \), we have \( b_4 < 2 \sum_{\ell \in [5,n]} b_\ell \) and thus

\[
\sum_{\ell \in [4,n]} b_\ell = b_4 + \sum_{\ell \in [5,n]} b_\ell \leq 3 \sum_{\ell \in [5,n]} b_\ell.
\]

(5.5)

By Lemma 4.2, the total overlapping time between \( M_1 \) and \( M_2 \) is

\[
o \geq \min \left\{ b_1, \sum_{\ell \in [2,3]} p_\ell \right\} + \min \left\{ b_4, \sum_{\ell \in [5,n]} p_\ell \right\} \\
\geq b_1 + \frac{1}{3} \sum_{\ell \in [4,n]} b_\ell = b_1 + \frac{1}{3} \left( b - \sum_{\ell \in [3]} b_\ell \right)
\]
where the second inequality is because of \( b_1 \leq b_2 + b_3 \), \( b_4 > \frac{1}{3} \sum_{\ell \in [4,n]} b_\ell \), and Eq. (3.5); the third inequality is due to \( b_2 + b_3 \leq \frac{3}{2} b_1 \). By Theorem 3.1 we have \( C^*_\text{max} \geq LB \geq \max \{ \frac{a+b}{2}, a \} \), and thus

\[
C^*_\text{max} = \frac{1}{3} a + \frac{2}{3} (a + b) = \frac{1}{3} a + \frac{2}{3} (a + b) - \frac{1}{6} b_1
\]

By Lemma 4.2, the total overlapping time between \( M_1 \) and \( M_2 \) is

\[
o \geq \min \{ b_1, p_2 \} + \min \left \{ b_3, \sum_{\ell \in [4,n]} p_\ell \right \}
\]

\[
\geq b_2 + b_4 \geq b_2 + \frac{2}{3} \sum_{\ell \in [4,n]} b_\ell = b_2 + \frac{2}{3} \left( b - \sum_{\ell \in [3]} b_\ell \right)
\]

where the second inequality follows from \( b_1 \geq b_2 \geq \ldots \geq b_n \); the third inequality is because of \( b_4 \geq \frac{2}{3} \sum_{\ell \in [4,n]} b_\ell \). By Theorem 3.1 we have \( C^*_\text{max} \geq LB \geq \max \{ \frac{a+b}{2}, a \} \), and thus

\[
C^*_\text{max} = a + b - o = a + b - \left( b_2 + \frac{2}{3} \left( b - \sum_{\ell \in [3]} b_\ell \right) \right)
\]

\[
= \frac{2}{3} a + \frac{1}{3} (a + b) + \frac{2}{3} b_1 - \frac{1}{3} b_2 + \frac{2}{3} b_3
\]

\[
\leq \frac{2}{3} a + \frac{1}{3} (a + b) + \frac{2}{3} b_1 + \frac{1}{3} b_3
\]

\[
\leq \frac{2}{3} a + \frac{1}{3} (a + b) + \frac{2}{3} b_1 + \frac{1}{3} \cdot \frac{3}{4} b_1 = \frac{2}{3} a + \frac{1}{3} (a + b) + \frac{11}{12} b_1
\]

\[
\leq \frac{2}{3} LB + \frac{1}{3} \cdot 2LB + \frac{11}{12} \cdot \frac{5}{16} LB = \frac{311}{192} LB \leq \frac{311}{192} C^*_\text{max} < \frac{13}{8} C^*_\text{max}
\]

where the first inequality is due to \( b_2 \geq b_3 \); the second inequality follows from \( b_3 \leq \frac{1}{2} (b_2 + b_3) \leq \frac{1}{2} \cdot \frac{3}{2} b_1 = \frac{3}{4} b_1 \); the third inequality is because of \( b_1 \leq \frac{5}{16} LB \).

**Case 5.2** \( b_2 + b_3 > \frac{3}{2} b_1 \): we consider the schedule \( \pi_6 \), which is \( \text{ALG1}(1,2,2) \) followed by \( \text{ALG1}(3,4,n) \). By Lemma 4.2 the total overlapping time between \( M_1 \) and \( M_2 \) is

\[
o \geq \min \{ b_1, p_2 \} + \min \left \{ b_3, \sum_{\ell \in [4,n]} p_\ell \right \} \geq b_2 + \min \left \{ b_3, \sum_{\ell \in [4,n]} p_\ell \right \}
\]
By Theorem 3.1, we have $C^*_{\text{max}} \geq LB \geq \max\{\frac{a+b}{2}, a\}$. If $b_3 \leq \sum_{\ell \in [4,n]} p_\ell$, by Lemma 4.1 we have

$$C^*_{\text{max}} = a + b - o \leq a + b - (b_2 + b_3) \leq a + b - \frac{3}{2} b_1 \leq 2LB - \frac{3}{2} \cdot \frac{2}{8}LB = \frac{13}{8}LB \leq \frac{13}{8}C^*_{\text{max}},$$

where the third inequality is because of $b_1 > \frac{2}{5}LB$. If $b_3 > \sum_{\ell \in [4,n]} p_\ell$, we have

$$C^*_{\text{max}} = a + b - o \leq a + b - b_2 - \sum_{\ell \in [4,n]} b_\ell = a + b_1 + b_3 \leq a + 2b_1 \leq LB + 2 \cdot \frac{5}{16}LB = \frac{13}{8}LB \leq \frac{13}{8}C^*_{\text{max}},$$

where the third inequality is because of $b_1 \leq \frac{5}{16}LB$.

To summarize, $\min\{C^*_{\text{max}}, C^*_{\text{max}}, C^*_{\text{max}}\} \leq \frac{13}{8}C^*_{\text{max}}$ holds in Case 5. This proves the lemma. \qed

**Lemma 5.6** In Case 6, i.e. $\frac{5}{16}LB \leq b_1 < \frac{3}{2}LB$ and the critical position $k \leq 3$, we have $\min\{C^*_{\text{max}}, C^*_{\text{max}}, C^*_{\text{max}}, C^*_{\text{max}}, C^*_{\text{max}}, C^*_{\text{max}}\} \leq \frac{13}{8}C^*_{\text{max}}$.

**Proof.** By the definition of the critical job and the fact that $b_1 \geq b_2 \geq \ldots \geq b_n$, we have $b_1 \leq b_2 + b_3 \leq 2b_1$. Similar to Case 5, we discuss two subcases according to the value of $b_2 + b_3$.

**Case 6.1** $b_2 + b_3 \leq \frac{6}{5}b_1$: we investigate two subcases based on the value of $\sum_{\ell \in [4,n]} b_\ell$.

- **Case 6.1.1** $\sum_{\ell \in [4,n]} b_\ell > \frac{2}{3}b_1$: we further investigate three subcases based on the value of $b_4$.

  - **Case 6.1.1.1** $b_4 \leq \frac{4}{3} \sum_{\ell \in [4,n]} b_\ell$:

    In this subcase, we consider the schedule $\pi_4$, which is $\text{ALG1}(1,2,3)$ followed by $\text{ALG2}(4,n)$. As $b_4 \leq \frac{4}{3} \sum_{\ell \in [4,n]} b_\ell$, Eq.(5.4) also holds in this case. That is,

    $$3 \cdot \sum_{\ell \in [5,n]} b_\ell \geq 2 \cdot \sum_{\ell \in [4,n]} b_\ell.$$

    By Lemma 4.2 and Lemma 4.3, the total overlapping time between $M_1$ and $M_2$ is

    $$o \geq \min \left\{ b_1, \sum_{\ell \in [2,3]} p_\ell \right\} + \frac{1}{2} \sum_{\ell \in [5,n]} b_\ell \geq b_1 + \frac{1}{2} \sum_{\ell \in [5,n]} b_\ell \geq b_1 + \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{2}{3}b_1 = \frac{11}{9}b_1 \geq \frac{55}{144}LB,$$

    where the second inequality follows from $b_1 \leq b_2 + b_3$; the third inequality is because of Eq.(5.4); the fourth inequality is because of $\sum_{\ell \in [4,n]} b_\ell > \frac{2}{3}b_1$; the last inequality is due to $b_1 \geq \frac{5}{16}LB$. By Theorem 3.1, we have $C^*_{\text{max}} \geq LB \geq \frac{a+b}{2}$, and thus

    $$C^*_{\text{max}} = a + b - o = a + b - \frac{55}{144}LB.$$
\[ 2LB - \frac{55}{144}LB = \frac{333}{144}LB \leq \frac{13}{8}C^*_\text{max}. \]

- **Case 6.1.1.2** \( b_4 \in \left( \frac{1}{3} \sum_{\ell \in [4,n]} b_\ell, \frac{2}{3} \sum_{\ell \in [4,n]} b_\ell \right) \): In this subcase, we consider the schedule \( \pi_5 \), which is ALG1(1, 2, 3) followed by ALG1(4, 5, n). As \( b_4 \leq \frac{2}{3} \sum_{\ell \in [4,n]} b_\ell \), Eq. (5.5) also holds in this case. That is,

\[ \sum_{\ell \in [4,n]} b_\ell \leq 3 \sum_{\ell \in [5,n]} b_\ell. \]

By Lemma 4.2, the total overlapping time between \( M_1 \) and \( M_2 \) is

\[ o \geq \min \left\{ b_1, \frac{1}{3} \sum_{\ell \in [2,3]} p_\ell \right\} + \min \left\{ b_4, \frac{1}{3} \sum_{\ell \in [5,n]} p_\ell \right\} \geq b_1 + \frac{1}{3} \sum_{\ell \in [4,n]} b_\ell \geq b_1 + \frac{1}{3} \cdot \frac{2}{3} b_1 = \frac{11}{9} b_1 \geq \frac{55}{144}LB, \]

where the second inequality is because of \( b_1 \leq b_2 + b_3 \) and Eq. (5.5); the third inequality is due to the condition \( \sum_{\ell \in [4,n]} b_\ell > \frac{2}{3} b_1 \) in Case 6.1.1; the last inequality is due to \( b_1 \geq \frac{5}{16}LB \). Similar to Case 6.1.1.1, we have \( C_{\text{max}}^\pi \leq \frac{12}{8}C^*_\text{max}. \)

- **Case 6.1.1.3** \( b_4 \geq \frac{2}{3} \sum_{\ell \in [4,n]} b_\ell \): As \( b_1 \geq b_2 \geq \ldots \geq b_n \) and the condition \( \sum_{\ell \in [4,n]} b_\ell > \frac{2}{3} b_1 \) in Case 6.1.1, we have

\[ b_3 \geq b_4 \geq \frac{2}{3} \sum_{\ell \in [4,n]} b_\ell \geq \frac{2}{3} \cdot \frac{2}{3} b_1 = \frac{4}{9} b_1. \] (5.6)

When \( a_2 \geq a_3 \), we consider the schedule \( \pi_8 \), which is ALG1(1, 4, n) followed by ALG1(2, 3, 3). When \( a_2 < a_3 \), we consider the schedule \( \pi_9 \), which is ALG1(1, 4, n) followed by ALG1(3, 2, 2). Due to the symmetry between \( \pi_8 \) and \( \pi_9 \), the analysis for the case \( a_2 < a_3 \) will be similar to the case \( a_2 \geq a_3 \). Next, we focus on analyzing \( \pi_8 \) when \( a_2 \geq a_3 \).

In this case, using Lemma 4.2 and Lemma 4.3 to estimate the lower bound of the overlapping processing time \( o \) between \( M_1 \) and \( M_2 \) is not enough. We need to consider the overlapping processing time caused by the concatenation of ALG1(1, 4, n) and ALG1(2, 3, 3). Refer to Figure 3 for details.

When \( \sum_{\ell \in [4,n]} p_\ell + a_2 \geq b_1 \), \( M_1 \) is always busy (refer to the left subfigure in Figure 3) and thus the total overlapping time between \( M_1 \) and \( M_2 \) is \( o \geq b_1 + \min\{b_2, p_3\} = b_1 + b_3 \).

By Lemma 4.1

\[ C_{\text{max}}^\pi = a + b - o \leq a + b - (b_1 + b_3) \leq a + b - \frac{13}{9} b_1 \]

\[ \leq 2LB - \frac{13}{9} \cdot \frac{5}{16}LB = \frac{233}{144}LB \leq \frac{13}{8}C^*_\text{max}, \]

where the second inequality is because of Eq. (5.6); the third inequality is due to \( b_1 \leq \frac{5}{16}LB \).
When $\sum_{\ell\in[4,n]} p_\ell + a_2 < b_1$ (refer to the right subfigure in Figure 3), the makespan can be computed as

$$C_{\max}^{\pi_8} = p_1 + \max\{p_3, b_2\} = p_1 + \max\{b_3 + a_3, b_2\} \leq p_1 + a_3 + b_2.$$ 

By Eq. (5.6), we have

$$b_1 > \sum_{\ell\in[4,n]} p_\ell + a_2 \geq b_4 + a_2 \geq \frac{4}{9}b_1 + a_2,$$

from which we can obtain $a_2 \leq \frac{5}{9}b_1$.

$$C_{\max}^{\pi_8} \leq p_1 + a_3 + b_2 \leq p_1 + \frac{5}{9}b_1 + b_2 \leq LB + \frac{5}{9} \cdot \frac{3}{8} LB + \frac{3}{8} LB = \frac{19}{12} LB \leq \frac{13}{8} C_{\max}^*,$$

where the second inequality is because of $a_3 \leq a_2 \leq \frac{5}{9}b_1$; the second inequality is due to $b_2 \leq b_1 < \frac{3}{8}LB$.

---

**Figure 3:** An illustration for $\pi_8 = \text{AlgL}1(1,4,n)$ followed by $\text{AlgL}1(2,3,3)$. The cases $\sum_{\ell\in[4,n]} p_\ell + a_2 \geq b_1$ and $\sum_{\ell\in[4,n]} p_\ell + a_2 < b_1$ are shown on the left and right respectively. Note that $p_3 < b_2$ may happen, which is not reflected in the figure.

**• Case 6.1.2** $\sum_{\ell\in[4,n]} b_\ell \leq \frac{2}{3}b_1$: In this subcase, we consider the schedule $\pi_6$, which is $\text{AlgL}1(1,2,2)$ followed by $\text{AlgL}1(3,4,n)$. By Lemma 4.2, the total overlapping time between $M_1$ and $M_2$ is

$$o \geq \min\{b_1, p_2\} + \min\left\{b_3, \sum_{\ell\in[4,n]} p_\ell\right\} \geq b_2 + \min\left\{b_3, \sum_{\ell\in[4,n]} p_\ell\right\},$$

where the second inequality follows from $b_1 \geq b_2 \geq \ldots \geq b_n$. Because $b_2 + b_3 \leq \frac{6}{5}b_1$, we have $b_3 \leq \frac{1}{2}(b_2 + b_3) \leq \frac{3}{5}b_1$. By Lemma 4.1, the makespan can be calculated as

$$C_{\max}^{\pi_6} = a + b - o \leq a + b - b_2 - \min\left\{b_3, \sum_{\ell\in[4,n]} p_\ell\right\}$$

$$= \max\left\{a + b - b_2 - b_3, a + b - b_2 - \sum_{\ell\in[4,n]} p_\ell\right\}$$
Case 6.2 $b_2 + b_3 > \frac{6}{5}b_1$: we investigate two subcases based on whether $a_1$ is equal to $\max\{a_1, a_2, a_3\}$.

- **Case 6.2.1** $\max\{a_1, a_2, a_3\} \neq a_1$:
  
  Similar to Case 6.1.1.3, we consider the schedule $\pi_8$ and $\pi_9$ when $a_2 \geq a_3$ and $a_2 < a_3$ respectively. Again, due to the symmetry between these two subcases, we focus on analyzing $\pi_8$ when $a_2 \geq a_3$. Recall that $\pi_8$ is $\text{ALG1}(1, 4, n)$ followed by $\text{ALG1}(2, 3, 3)$. A visualization demo can be found in Figure 3.

When $\sum_{\ell \in [4, n]} p_\ell + a_2 \geq b_1$, $M_1$ is always busy (refer to the left subfigure in Figure 3) and thus the total overlapping time between $M_1$ and $M_2$ is $o \geq b_1 + \min\{b_2, p_3\} = b_1 + b_3$. Therefore, we have

$$C_{\pi_8}^{\max} = a + b - o \leq a + b - (b_1 + b_3) \leq a + b - (b_2 + b_3)$$

$$\leq a + b - \frac{6}{5}b_1 \leq 2LB - \frac{6}{5} \cdot \frac{5}{10}LB = \frac{13}{8}LB \leq \frac{13}{8}C_{\max}^*,$$

where the second inequality is because of $b_1 \geq b_2$; the third inequality is due to $b_2 + b_3 > \frac{6}{5}b_1$; the fourth inequality follows from $b_1 \geq \frac{5}{16}LB$.

When $\sum_{\ell \in [4, n]} p_\ell + a_2 < b_1$ (refer to the left subfigure in Figure 3), we have $a_2 \leq b_1$ and the makespan can be computed as

$$C_{\pi_8}^{\max} = p_1 + \max\{p_3, b_2\} \leq p_1 + a_3 + b_2 = a_1 + b_1 + a_3 + b_2$$

$$\leq 4b_1 \leq 4 \cdot \frac{3}{8}LB \leq \frac{13}{8}LB \leq \frac{13}{8}C_{\max}^*,$$

where the second inequality is because of $b_2 \leq b_1$ and $a_1 \leq \max\{a_2, a_3\} = a_2 \leq b_1$.

- **Case 6.2.2** $\max\{a_1, a_2, a_3\} = a_1$: In this subcase, we consider the schedule $\pi_7$, which is $\text{ALG1}(3, 4, n)$ followed by $\text{ALG1}(1, 2, 2)$.

When $\sum_{\ell \in [4, n]} p_\ell + a_1 \geq b_3$ (refer to the left subfigure in Figure 4), $M_1$ is always busy and thus the total overlapping time between $M_1$ and $M_2$ is

$$o \geq b_3 + \min\{b_1, p_2\} \geq b_3 + b_2 \geq \frac{6}{5}b_1.$$ 

By Lemma 4.1, we have

$$C_{\max}^{\pi_7} = a + b - o \leq a + b - \frac{6}{5}b_1.$$
\[
\begin{align*}
\leq  2LB + \frac{6}{5} \cdot \frac{5}{16}LB &= \frac{13}{8}LB \leq \frac{13}{8}C^*_{\text{max}}.
\end{align*}
\]

When \( \sum_{\ell \in [4,n]} p_\ell + a_1 < b_3 \), (refer to the right subfigure in Figure 4), the makespan is computed as
\[
C^\pi_{\text{max}} = p_3 + \max\{b_1, p_2\} \leq a_3 + b_3 + b_1 + a_2 \leq 2(a_1 + b_1)
\leq 4b_1 \leq 4 \cdot \frac{3}{8}LB = \frac{3}{2}LB \leq \frac{13}{8}LB \leq \frac{13}{8}C^*_{\text{max}},
\]

where the second inequality is because of \( \max\{a_1, a_2, a_3\} = a_1 \) and \( b_1 \geq b_2 \geq b_3 \); the third inequality is due to \( a_1 \leq \sum_{\ell \in [4,n]} p_\ell + a_1 < b_3 \leq b_1 \).

To summarize, \( \min\{C^\pi_{\text{max}}, C^\pi_5_{\text{max}}, C^\pi_6_{\text{max}}, C^\pi_{\pi_7}, C^\pi_{\pi_8}, C^\pi_{\pi_9}\} \leq \frac{13}{8}C^*_{\text{max}} \) holds in Case 6. This proves the lemma.

\[\square\]

**Theorem 5.1** The Max-Overlap algorithm is a \( O(n) \)-time approximation algorithm with a worst-case performance ratio \( \frac{13}{8} \).

**Proof.** We consider the brute-force version, which simply computes 9 feasible schedules and returns the one with the minimum makespan. Without loss of generality, we assume all operations are integers. Sorting jobs with respect to the size of the second-stage operations takes linear time if any linear sorting algorithm, such as Radix-Sort, is applied. Computing each schedule takes \( O(n) \) time. In total, the Max-Overlap algorithm has a linear time complexity.

Lemmas 5.1-5.6 consider all possible inputs and guarantee to find a feasible schedule with makespan at most \( \frac{13}{8}C^*_{\text{max}} \). In other words, among the 9 schedules \( \{\pi_1, \pi_2, \ldots, \pi_9\} \), no matter what job sequence is taken as an input, there always exists one schedule \( \pi \) such that \( C^\pi_{\text{max}} \leq \frac{13}{8}C^*_{\text{max}} \). Therefore, the approximation ratio of the Max-Overlap algorithm is \( \frac{13}{8} \).

\[\square\]
6 Conclusions

In this paper, we study the $F_2|\text{nwt,mtflx} M_1 \rightarrow M_2|C_{\text{max}}$ problem, a variant of the classic two-stage flowshop scheduling problem with the makespan minimization objective. We add the no-wait constraint and allow the first-stage machine to have the multi-task flexibility. Both constraints are motivated by the arrangement of two most intensive resources in hospitals, i.e. operation room beds and recovery room beds. In addition, our problem is capable of modeling more real-world applications, such as robot cell scheduling containing CNC machines or PCBs. We investigate the NP-hardness of the proposed problem and design a linear-time algorithm achieving an approximation ratio of $\frac{13}{8}$. Our algorithm utilizes both no-wait and multi-task flexibility to compress the feasible schedule such that the overlapping processing time between two machines is as large as possible. The analysis of our algorithm is based on case-by-case discussions by using the nontrivial structural properties of the problem we discovered. We believe the approximation ratio cannot have a big improvement under the current ideas and structural properties. In future, we will follow these three directions: 1) discover more novel structural properties to increase the approximation ratio; 2) extend the current model for more real-applications to introduce more interesting models and utilize the current idea to design practical algorithms; 3) design high performance heuristic algorithms for the problem and conduct simulation and/or numerical study.

Acknowledgements

Dong is supported by the National Natural Science Foundation of China Grants No. 11971435 and the Zhejiang Provincial Natural Science Foundation Grants No. LY18A010029.

References

[1] T. Aldowaisan, A. Allahverdi, New heuristics for no-wait flowshops to minimize makespan, Computers & Operations Research 30(2003) 1219-1231.

[2] A. Allahverdi, H. Aydilek, A. Aydilek, No-Wait Flowshop Scheduling Problem with two Criteria; Total Tardiness and Makespan, European Journal of Operational Research 269(2018) 590-601.

[3] A. Allahverdi, M. Allahverdi, Two-machine no-wait flowshop scheduling problem with uncertain setup times to minimize maximum lateness, Computational and Applied Mathematics 37(2018) 6774-6794.

[4] A. Allahverdi, A survey of scheduling problems with no-wait in process, European Journal of Operational Research 255 (2016) 665-686.
[5] J. Browne, J. Harhen, J. Shivnan, Production Management Systems. Addison-Wesley, New York, 1996.

[6] T.C.E. Cheng, G. Wang, A note on scheduling alternative operations in two-machine flowshops, *Journal of the Operational Research Society* 49(6) (1998) 670-673.

[7] Y. Crama, H. Gultekin, Throughput optimization in two-machine flowshops with flexible operations. *Journal of Scheduling* 13 (2010) 227-243

[8] M.-L. Espinouse, P. Formanowicz, B. Penz, Complexity results and approximation algorithms for the two machine no-wait flow-shop with limited machine availability. *Journal of the Operational Research Society* 52(1) (2001) 116-121.

[9] M.-L. Espinouse, P. Formanowicz, B. Penz, Minimizing the makespan in the two-machine no-wait flow-shop with limited machine availability. *Computers and Industrial Engineering* 37(1) (1999) 497-500.

[10] P.C. Gilmor, R.E. Gomory, Sequencing a One State-Variable Machine: A Solvable Case of the Traveling Salesman Problem, *Operations Research* 12(5) (1964) 655-679.

[11] C. A. Glass, J. N. D. Gupta, C. N. Potts, Two-machine no-wait flow shop scheduling with missing operations. *Mathematics of Operations Research* 24(4) (1999) 911-924.

[12] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.R. Kan, Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey, *Annals of Discrete Mathematics* 5(1) (1979) 287-326.

[13] N.G. Hall, C. Sriskantharajah, A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process, *Operations Research* 44(3) (1996) 510-525.

[14] A. Jebali, A. B. Hadj Alouane, P. Ladet, Operating rooms scheduling, *International Journal of Production Economics* 99(1) (2006) 52-62.

[15] V. Kats, E. Levner, Cyclic scheduling in a robotic production line. *Journal of Scheduling* 5 (2002) 23-41.

[16] D. Khorasanian, G. Moslehi, Two-machine flow shop scheduling problem with blocking, multi-task flexibility of the first machine, and preemption, *Computers & Operations Research* 79 (2016) 94-108.

[17] E.-J. Lee, P. B. Mirchandani, Concurrent routing, sequencing, and setups for a two-machine flexible manufacturing cell. *IEEE journal of robotics and automation* 4(3) (1988) 256-264.

[18] X. Li, Z. Yang, R. Ruiz, T. Chen, S. Sui, An iterated greedy heuristic for no-wait flow shops with sequence dependent setup times, learning and forgetting effects, *Information Sciences* 453(2018) 408-425.
[19] C.J. Liao, C.L. Sun, W.C. You, Flow-shop scheduling with flexible processors, Computers & Operations Research 22(3) (1995) 297-306.

[20] B. L. Maccarthy, J. Liu, Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling, International Journal of Production Research 31(1) (1993) 59C79.

[21] S. Martinez, S. Dauzre-Prs, C. Guret, Y. Mati, N. Sauer, Complexity of flowshop scheduling problems with a new blocking constraint. European Journal of Operational Research 169(3) (2006) 855C864.

[22] H. H. Miyata, M. S. Nagano, The blocking flow shop scheduling problem: A comprehensive and conceptual review Expert Systems With Applications 137 (2019) 130C156.

[23] Q. K. Pan, M. F. Tasgetiren, Y.C. Liang, A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem, Computers and Operations Research 35(2008) 2807-2839.

[24] Q. K. Pan, L. Wang, B. H. Zhao, An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion, The International Journal of Advanced Manufacturing Technology 38(2008) 778-786.

[25] C.H. Pan, J.S. Chen, Scheduling alternative operations in two-machine flow-shops, Journal of the Operational Research Society 48(5) (1997) 533-540.

[26] J. Piehler, Ein Beitrag zum Reihenfolgeproblem, Mathematical Methods of Operations Research (ZOR) 4(3) (1960) 138-142.

[27] W. Qi, H. Yong, A two-stage semi-hybrid flowshop problem in graphics processing, Applied Mathematics-A Journal of Chinese Universities 20(4) (2005) 393-400.

[28] C. Rajendran, A No-Wait Flowsheet Scheduling Heuristic to Minimize Makespan,The Journal of the Operational Research Society 45(1994) 472-478.

[29] S. S. Reddi, C. V. Ramamoorthy, On the flow-shop sequencing with no wait in process. Operations Research Quarterly 23(3) (1972) 323C328.

[30] H. Röck, The Three-Machine No-Wait Flow Shop is NP-Complete, Journal of the ACM 31(2) (1984) 336-345.

[31] R. Tavakkoli-Moghaddam, A. Rahimi-Vahed, A.H. Mirzaei, A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: Weighted mean completion time and weighted mean tardiness, Information Sciences 177(22) (2007) 5072-5090.

[32] S. Wang, M. Liu, C. Chu, A branch-and-bound algorithm for two-stage no-wait hybrid flowshop scheduling, International Journal of Production Research 53(2015) 1143-1167.
[33] T. Y. Wang, Y.-H. Yang, H.-J. Lin, Comparison of scheduling efficiency in two/three-machine no-wait flow shop problem using simulated annealing and genetic algorithm. *Asia-Pacific Journal of Operational Research* 23(1) (2006) 41-59.

[34] Q. Wei, E. Shan, L. Kang, A FPTAS for a two-stage hybrid flow shop problem and optimal algorithms for identical jobs, *Theoretical Computer Science* 524 (2014) 78-89.

[35] CDC (Centers for Disease Control and Prevention) of USA Coronavirus (COVID-19) https://www.cdc.gov/coronavirus/2019-ncov/index.html (2020).