# On the NP-Hardness of Two Scheduling Problems Under Linear Constraints

Kameng Nip(✉)

School of Mathematical Sciences, Xiamen University, Xiamen, China
kmnip@xmu.edu.cn

**Abstract.** In this work, we investigate the computational complexity of two different scheduling problems under linear constraints, including single-machine scheduling problem with total completion time and no-wait two-machine flow shop scheduling problem. In these problems, a set of jobs must be scheduled one or more machines while the processing times of them are not fixed and known in advance, but are required to be determined by a system of given linear constraints. The objective is to determine the processing time of each job, and find the schedule that minimizes a specific criterion, e.g., makespan or total completion time among all the feasible choices. Although the original scheduling problems are polynomially solvable, we show that the problems under linear constraints become NP-hard. We also propose polynomial time exact or approximation algorithms for various special cases of them. Particularly, we show that when the total number of constraints is a fixed constant, both problems can be solved in polynomial time by utilizing the scheduling algorithms and the properties of linear programming.

**Keywords:** Scheduling · linear programming · computational complexity

## 1 Introduction

In the presented work, we study two different machine problems under linear constraints, including single-machine scheduling problem with minimizing total completion time $1||\sum_j C_j$ and no-wait two-machine flow shop scheduling problem $F2|no-wait|C_{\max}$. In scheduling problem under linear constraint problems, the processing times of jobs are not fixed and known beforehand, which distinguishes them from classic scheduling problems and adds an additional layer of flexibility to the decision-making process. Instead, the decision maker only knows the information that the processing times satisfy a system of given linear constraints. The goal is to determine the processing time of each job, and the

This research work is partially supported by the Natural Science Foundation of Fujian Province of China No. 2021J05011 and the Fundamental Research Funds for the Central Universities of Xiamen University No. 20720210033.

© Springer Nature Switzerland AG 2023

schedule to one or more machines such that certain objective, e.g., the makespan $C_{\max}$ or the total completion time $\sum_j C_j$ is minimized.

The scheduling problem under linear constraints (SLC in short) was first introduced in [18], in which the machine environment is identical parallel machine and the objective is to minimize the makespan, that is, $P||C_{\max}$ where $C_{\max}$ is the completion time of the last job. More specifically, there are $k$ identical parallel machines and $n$ jobs to be processed, with a matrix $A \in \mathbb{R}^{k\times n}$ and a vector $\boldsymbol{b} \in \mathbb{R}^{k\times 1}$. The processing times $\boldsymbol{x} = (x_1, ..., x_n) \in \mathbb{R}^n$ are also decision variables that have to satisfy $A\boldsymbol{x} \geq \boldsymbol{b}$ and $\boldsymbol{x} \geq 0$. The objective of the SLC problem is to find the values of processing times $\boldsymbol{x}$ as well as the schedule of jobs to the machines, which leads to the minimum makespan among all the feasible choices. Note that the SLC problem is a generalization of the original scheduling problem, since one can reduce the original one to the SLC problem by setting the linear constraints as $x_1 = p_1$, $x_2 = p_2$, ..., $x_n = p_n$, where $p_i$ is the fixed processing times of the problem without linear constraints. Moreover, the authors showed that if the number of linear constraints $k$ is a fixed constant, then the SLC problem can be solved in polynomial time through searching the basic feasible solutions of a series of linear programs. They also proposed several approximation algorithms for the general case where $k$ is an input of problem instance. It is worth-noting that the original parallel machine scheduling problem $P||C_{\max}$ is a widely recognized NP-hard problem even for the case of two machines, and is strongly NP-hard in general [4]. Subsequently, several scheduling problems that involve processing times or other parameters (such as machine speeds) satisfying linear constraints have been investigated in this research direction. These problems includes uniformly parallel machines [27,28], the two-machine flow shop machine and other shop machine environments [16]. The above results illustrate a sharp difference in computational complexity between the original scheduling problems and those under linear constraints. For instance, the two-machine flow shop scheduling problem $F2||C_{\max}$ can be solved in polynomial time by Johnson's rule [10], while the two-machine flow shop scheduling problem under linear constraints (2-FLC in short) is NP-hard in the strong sense as shown in [15]. In comparison, the two-machine open shop scheduling problem under linear constraints (2-OLC in short) can be solved in polynomial time [16], which has the similar computational complexity as the original two-machine open shop scheduling problem $O2||C_{\max}$ [6]. Additionally, there have been recent research on different other combinatorial optimization problems with linear constraints, such as bin packing problem [24], knapsack problem [19], and various graph optimization problems [14,17]. The findings of these studies prompt further investigation into the computational complexity and algorithmic designs for problems under linear constraints. Such research could potentially offer theoretically benefits to the broader field combinatorial optimization as well.

As indicated in previous works, the scheduling problems under linear constraints could have potential practical applications beyond their theoretical interest. To motivate the two problems discussed in this work, we present some application scenarios as follows.

– *Consumer service.* The single machine scheduling problem, which minimizes total completion time, has a wide range of applications in service systems, such as hospitals, restaurants, and banks to enhance consumer satisfaction [21]. The processing time of each job can be viewed as a service time of each consumer, and the machine can be viewed as a service producer. The common objective to the service producer is to decide the serving schedule that can minimize the mean total waiting and service time of the consumers, which is closely related to the total completion time (also called mean flow time in the literature) of jobs in the machine. In the simplest and offline setting, the service producer possesses complete information regarding the service time required for each consumer, say, those make reservations for the upcoming business day. Therefore, it is optimal to schedule the service order according to the famous shortest processing time first rule [21], which is also known as Smith's rule [23], such that the mean total waiting time is minimized. However, in some situations, it is possible that the service time of each customer is not so accurately determined and could be flexible. For example, consider the scenario outlined in Table 1 for a concrete illustration. Assume

**Table 1.** Example for the application of consumer service.

| Constraints | Consumer | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | $\cdots$ | $n$ | |
| revenue | $100x_1$ | $200x_2$ | $\cdots$ | $120x_n$ | $\geq 10000$ |
| resource A | $3x_1$ | $5x_2$ | $\cdots$ | $7x_n$ | $\leq 30$ |
| resource B | $5x_1$ | $10x_2$ | $\cdots$ | $6x_n$ | $\leq 50$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| min. of 1 | $x_1$ | 0 | $\cdots$ | 0 | $\geq 10$ |
| max. of 1 | $x_1$ | 0 | $\cdots$ | 0 | $\leq 20$ |
| min. of 2 | $x_2$ | 0 | $\cdots$ | 0 | $\geq 15$ |
| max. of 2 | $x_2$ | 0 | $\cdots$ | 0 | $\leq 30$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| max. of $n$ | 0 | 0 | $\cdots$ | $x_n$ | $\leq 15$ |

that we have $n$ consumer to be served, and $x_i$ is the service time of consumer $i$. Each consumer has an interval of her acceptable service time based on her own situation, e.g., $10 \leq x_1 \leq 20$ for consumer 1, which indicates that the appropriate time to serve this consumer would be between 10 and 20 unit times. Moreover, each unit of service time will generate a specific amount of profit or utility to the service producer. For example, if the service time of consumer 1, 2, ..., $n$ are $x_1$, ..., $n$, respectively, then the service producer would receive revenues of $100x_1$, $200x_2$, ..., $120x_n$, respectively. The service producer aims to gain a total revenue of at least 10000, which naturally leads to $100x_1 + 200x_2 + \cdots 120x_n \geq 10000$. Moreover, the service typically requires the consumption of various resources, including human labor, electricity, computational resources, and more. Particularly in Table 1, each unit service time of consumer 1 requires 3 unit amounts of resource $A$, 5 unit amounts for consumer 2, and so forth. In other words, the processing time of jobs should also satisfy the linear constraint $3x_1 + 5x_2 + \cdots 7x_n \leq 30$ for the limit of resource $A$, and so for the other resources. The service provider needs to assign service times to consumers based on linear constraints and schedule their service to minimize the total completion time.

– *Industrial Production.* Scheduling problems have a wide range of applications in industrial production. In previous works [15,16,18], the authors presented various practical scenarios regarding the scheduling problems under linear constraints. For instance, the 2-FLC problem is motivated by steel manufacturing [15,16]. The decision maker wants to obtain specific quantities of several raw metals by extracting them from multiple types of steel. Each type of steel (iron, copper, aluminum, and etc.) corresponds to a job. The restrictions such as requirements of the raw metals corresponds to the linear constraints of the processing times. There are different essential steps in the flow-shop production process, such as wire-drawing and annealing, and must be completed in that order for each job. The decision maker aims to determine the processing time of each job while consider linear constraints, and to schedule the jobs on the flow-shop machines to minimize the makespan.
  In certain scenarios, such as chemical processing, food processing, automobile assembly and working planning [3,8], it is necessary for a job to start processing on the second machine immediately after completing its operations on the first machine. For instance, if the materials are not cooled down quickly, it can cause undesirable issue such as transforming into some other substances. Therefore, it is practical to extend the above 2-FLC problem to the machine environment that the jobs satisfy no-wait restrictions.

In this work, we focus on investigating the computational complexity of the two scheduling problems under linear constraints mentioned above, namely, single-machine scheduling problem under linear constraints (with minimizing total completion time) and no-wait two-machine flow shop scheduling problem under linear constraints. We show that both two problems are NP-hard. It is worth noting that the original problems $1||\sum_j C_j$ and $F2|no-wait|C_{\max}$ can be solved in polynomial time, as we will review in later sections. Our findings indicate that solving the problems under linear constraints is more computationally difficult than solving their original counterparts. Moreover, we propose approximation algorithms for the general cases, which are obtained by solving some specific linear programs. Then we consider several nontrivial special cases of them. In particular, we show that if the total number of fixed constraints is a constant, then both problems can be solved in polynomial time. The algorithms are based on the properties of basic feasible solutions for linear programming and their scheduling algorithms.

The remainder of this work is organized as follows. In Sect. 2, we study the single machine scheduling problem under linear constraints, while in Sect. 3, we study the no-wait two-machine flow shop scheduling problem under linear constraints. For each section, we first formally define the corresponding problem and briefly review the related literature. Then, we analyze the computational complexity, and present polynomial-time optimal or approximation algorithms for various cases. Finally, in Sect. 4, we provide some concluding remarks.

## 2 Single Machine Scheduling Under Linear Constraints

### 2.1 Problem Definition and Literature Review

The single machine scheduling problem under linear constraints (SSLC problem in short) is formally defined as follows.

**Definition 1.** *Given $n$ jobs and a single machine. Each job $i$ has a processing time $x_i$, which are determined by $k$ linear constraints $A\boldsymbol{x} \geq \boldsymbol{b}$ with $A \in \mathbb{R}^{k\times n}$ and $\boldsymbol{b} \in \mathbb{R}^{k\times 1}$. The goal of the SSLC problem is to determine the processing times of the jobs such that they satisfy the linear constraints and to schedule the jobs to the machines to minimize the total completion time $\sum_j C_j$.*

The problem $1||\sum_j C_j$ is perhaps the simplest scheduling model studied in the field of scheduling. In [23], it is shown that the optimal schedule is to assign the jobs in a non-decreasing order of their processing jobs. Such rule is referred to Smith's rule or the shortest processing time first rule [21], and can be implemented in $O(n \log n)$ time. In other words, let $p_1 \leq p_2 \leq ... \leq p_n$ the (fixed) processing times of the $n$ jobs and $OPT$ be the optimal value, then the optimal total completion time for $1||\sum_j C_j$ is given by

$$OPT = np_1 + (n-1)p_2 + \cdots + 2p_{n-1} + p_n. \tag{1}$$

Smith [23] also extended the idea of Simth's rule to solve a more general model $1||\sum_j w_j C_j$, in which each job has a nonnegative weight $w_j$ and the objective is to minimize the total weighted completion time. Later, [1] showed that a more complicated extension to the unrelated parallel machine environments $R||\sum_j C_j$ can be solved in polynomial time, by reducing it to the transportation problem. However, many slight extensions of single machine scheduling problem turn out to be NP-hard. For instance, [1] showed that $P2||\sum_j w_j C_j$ is NP-hard, and [12] proved that $1|r_j|\sum_j C_j$ is strongly NP-hard in which $r_j$ is the arrival time of job $j$. For more details on the machine scheduling problem with minimizing total completion time, we refer to the textbook [21] and some recent works [9,11]. For the scheduling problem under the linear constraints, [18] showed that the problem $1||C_{\max}$ with linear constraints is also polynomially solvable, by solving a linear program that minimizes the total processing time $\sum_{j=1}^{n} p_j$. Conversely, the SSLC problem turns out to be more difficult than the original scheduling problem $1||\sum_j C_j$, as we will demonstrate that it is indeed NP-hard.

## 2.2 Computational Complexity

By (1), the optimal schedule of $1||\sum_j C_j$ is to assign the jobs in a non-decreasing order of their processing times. Therefore, we can reformulate the SSLC problem as the following mathematical optimization problem (2):

$$\begin{aligned} \min \quad & n\Theta_1(\boldsymbol{x}) + (n-1)\Theta_2(\boldsymbol{x}) + \cdots + 2\Theta_{n-1}(\boldsymbol{x}) + \Theta_n(\boldsymbol{x}) && \text{(2a)} \\ \text{s.t.} \quad & A\boldsymbol{x} \geq \boldsymbol{b} && \text{(2b)} \\ & \boldsymbol{x} \geq 0, && \text{(2c)} \end{aligned}$$

where $\Theta_k : \mathbb{R}^n \to \mathbb{R}$ maps $\boldsymbol{x}$ to its $k$-largest element. In other words, we have $\Theta_1(\boldsymbol{x}) = \min_{i=1,\ldots,n}\{x_i\}$, $\Theta_n(\boldsymbol{x}) = \max_{i=1,\ldots,n}\{x_i\}$ and $\Theta_1(\boldsymbol{x}) \leq \Theta_2(\boldsymbol{x}) \leq \cdots \leq \Theta_n(\boldsymbol{x})$ where $\Theta_k(\boldsymbol{x})$ is the $k$-largest element of $\boldsymbol{x}$. We remark that the objective function is considered as a weighted average function in the literature [20, 25, 26]. Particularly, let $f(\boldsymbol{x}) = w_1\Theta_1(\boldsymbol{x}) + w_2\Theta_2(\boldsymbol{x}) \cdots + w_{n-1}\Theta_{n-1}(\boldsymbol{x}) + w_n\Theta_n(\boldsymbol{x})$. If the weights satisfy $w_1 \leq w_2 \leq \cdots \leq w_n$, then it has been shown that the nonlinear optimization problem $\min f(\boldsymbol{x})$ subject to (2b) and (2b) can be equivalently reformulated as a linear programming problem [20]. In other words, the corresponding problem can be solvable in polynomial time, e.g., by some ellipsoid method or interior point method [13]. However, the idea of transforming (2) into an equivalent linear programming formulation may not be a viable option when the weights follow the condition $w_1 \geq w_2 \geq \cdots \geq w_n$. To the best of our knowledge, the computational complexity of such linear programming problem with a non-increasing weighted average objective function is unclear. In the following, we show that the SSLC problem (2) is NP-hard, which significantly differs from the original scheduling problem $1||\sum_j C_j$, as well as other optimization problems involving a weighted average function.

**Theorem 1.** *The SSLC problem (2) is NP-hard.*

**Proof.** We reduce the independent set problem to the decision problem of the SSLC problem (2). The decision problem of independent set problem is given a graph $G = (V, E)$ and an integer $K$, to decide whether there is a vertex set $V' \in V$ with size at least $K$ in which no two vertices are adjacent. Let $n = |V|$ and $m = |E|$, we construct an instance of SSLC with $2n$ variables that has processing times denoted by $x_1, ..., x_n$ and $y_1, ..., y_n$, where $x_j$ and $y_j$ correspond to vertex $v_j$ in $V$, and $m + n + 1$ linear constraints:

$$\begin{aligned} \sum_{i=1}^{n} x_i &\geq K && \text{(3a)} \\ x_u + x_v &\leq 1, \quad \forall (u, v) \in E && \text{(3b)} \\ x_j + y_j &= 1, \quad \forall j = 1, ..., n && \text{(3c)} \\ \boldsymbol{x}, \boldsymbol{y} &\geq 0. && \text{(3d)} \end{aligned}$$

It suffices to show that there is an independent set with size at least $K$ if and only if there is a setting of processing times for the jobs that are feasible to (3) of the SSLC problem, and a corresponding schedule with a total completion time of no more than $\frac{n(n+1)}{2}$.

On one hand, if there is an independent set $V'$ with size at least $K$, then for each vertex $v_j \in V'$, we set $x_j = 1$ and $y_j = 0$; for $v_j \notin V'$, we set $x_j = 0$ and $y_j = 1$. The processing times of jobs are feasible to (3), since the jobs correspond to $V'$ form an independent set. Moreover, there are exactly $n$ jobs that have processing times 1, and the other $n$ jobs have processing times 0. The total completion time of this schedule of this instance is exactly $n + (n-1) + \cdots + 1 = \frac{n(n+1)}{2}$.

On the other hand, assume that there is an instance of the SSLC problem, in which the processing times $\boldsymbol{x}$ and $\boldsymbol{y}$ are feasible to (3), and a corresponding schedule $S$ with total completion time no more than $\frac{n(n+1)}{2}$. Let $p_1 \leq p_2 \leq \cdots \leq p_{n-1} \leq p_n$ be the $n$ largest values among the $2n$ variables $\boldsymbol{x}$ and $\boldsymbol{y}$ in this solution, where $p_j$ could be certain $x_i$ or $y_i$ for some $i$. Accordingly, for each $j = 1, ..., n$, we denote $\tilde{p}_j = y_i$ if $p_j = x_i$, and $\tilde{p}_j = x_i$ if $p_j = y_i$. Note that we have $x_j + y_j = 1$ for all $j$ by constraint (3c), and $1 - p_n \geq \cdots \geq 1 - p_1$ by the definition of $p_j$. In other words, we relabel the variables $x_1$, ..., $x_n$ and $y_1$, ..., $y_n$ by $p_1$, ..., $p_n$ and $\tilde{p}_1$, ..., $\tilde{p}_n$, which satisfies

$$\tilde{p}_n \leq \cdots \leq \tilde{p}_1 \leq p_1 \leq \cdots \leq p_n. \tag{4}$$

Next we claim that any feasible schedule of this instance must have total completion time at least $\frac{n(n+1)}{2}$, and the equality holds only if $p_1 = \cdots = p_n = 1$ and $\tilde{p}_1 = \cdots = \tilde{p}_n = 0$. To see this, we consider the best possible schedule of this instance, which is scheduled by the shortest processing time first/Smith's rule. Then from (1) and (4), its total completion time is given by

$$\begin{aligned} &2n\tilde{p}_n + (2n-1)\tilde{p}_{n-1} + \cdots + (n+1)\tilde{p}_1 + np_1 + \cdots + 2p_{n-1} + p_n \\ =& 2n(1-p_n) + p_n + (2n-1)(1-p_{n-1}) + 2p_{n-1} + \cdots + (n+1)(1-p_1) + np_1 \\ =& 2n + (2n-1) + \cdots + (n+1) + (1-2n)p_n + (3-2n)p_{n-1} + \cdots + (-1)p_1. \end{aligned}$$

Consider the following linear programming problem:

$$\begin{aligned} \min\ & (1-2n)p_n + (3-2n)p_{n-1} + \cdots + (-1)p_1 \\ \text{s.t.}\ & \qquad 0 \leq p_j \leq 1, \qquad \forall j = 1, ..., n \end{aligned} \tag{5}$$

Since all coefficients of its objective function are negative, we can verify that $p_1 = \cdots = p_n = 1$ is the unique optimal solution to (5). By definition, it follows that $\tilde{p}_1 = \cdots = \tilde{p}_n = 0$ and the claim is proved. Therefore, the total completion time of any feasible schedule to this instance is no less than the optiaml value to (5), namely, $n + (n-1) + \cdots + 1 = \frac{n(n+1)}{2}$. By assumption, the total completion time of the schedule in this SSLC instance is exactly $\frac{n(n+1)}{2}$, in which exactly $n$ variables among $\boldsymbol{x}$ and $\boldsymbol{y}$ are 1, and the other $n$ variables are 0. We can just select the vertex $v_j$ with $x_j = 1$ into $V'$, which constitutes an independent set with size at least $K$ by (3a) and (3b). It finishes the proof of the theorem. $\square$

### 2.3 Algorithms

The hardness result in Theorem 1 indicates that it is impossible to find an optimal solution to (2) in polynomial time unless $P = NP$. In this subsection, we develop algorithms for solving the SSLC problem. For the general case, we can obtain an $n$-approximation algorithm as described in Theorem 2. Due to the limitation of space, we omit the details of the subsequent lemmas and theorems. The details will be provided in the full version.

**Theorem 2.** *The SSLC problem has an $n$-approximation algorithm.*

It should be noted that the number of constraints $k$ in the instance of the hardness reduction in Theorem 1 is not a fixed constant. In the following, we consider a special case when $k$ is not fixed. We show that this case can be solved in polynomial time, which depends on the following property.

**Lemma 1.** *The SSLC problem has an optimal solution in which at most $k$ jobs have nonzero processing time.*

By Lemma 1, there exists an optimal solution that contains a constant number $k$ of nonzero processing time. Therefore, we can find the optimal solution by first enumerating all the nonzero processing time jobs and the optimal schedule, then solve a specific linear program to obtain the best processing times. The detail is summarized in Algorithm 1 and Theorem 3.

**Algorithm 1.** Enumeration algorithm for the SSLC problem with fixed $k$

```
1: for each subset $J'$ of $J$ with $k$ jobs do
2:     for each possible permutation of $J'$ do
3:         Let $(\sigma(1), ..., \sigma(k))$ be the permutation, solve the following linear program
           while setting $x_i = 0$ for $i \notin J'$:
```

$$\begin{aligned} \min\ & kx_{\sigma(1)} + (k-1)x_{\sigma(2)} + \cdots + x_{\sigma(k)} \\ \text{s.t.}\ & A\boldsymbol{x} \geq \boldsymbol{b} \\ & \boldsymbol{x} \geq 0. \end{aligned} \tag{6}$$

```
4:         if (6) is feasible then
5:             Let the processing times of jobs be the optimal solution to (6), and record
               the schedule and the makespan.
6: return the schedule with the smallest total completion time among all these iter-
   ations and its corresponding processing times.
```

**Theorem 3.** *Algorithm 1 returns an optimal solution to the SLC problem and has time complexity $O(n^k L)$, where the parameter $L$ is the input length of (6).*

## 3 No-Wait Two-Machine Flow Shop Scheduling Problem Under Linear Constraints

### 3.1 Problem Definition and Literature Review

The no-wait two-machine flow shop scheduling problem under the linear constraints (no-wait 2FLC problem in short) is formally defined as follows.

**Definition 2.** *Given $n$ jobs and two flow-shop machines. Each job has to be processed on the first machine and then on the second machine, and the process in the second machine must be start immediately after its finish in the first machine (no-wait restriction). The processing times of job $i$ on the first machine and the second machine are $x_i$ and $y_i$ respectively, which are determined by $k$ linear constraints, $A\boldsymbol{x} + C\boldsymbol{y} \geq \boldsymbol{b}$ with $A, C \in \mathbb{R}^{k\times n}$ and $\boldsymbol{b} \in \mathbb{R}^{k\times 1}$. The goal of the no-wait 2FLC problem is to determine the processing times of the jobs such that they satisfy the linear constraints and to schedule the jobs to the two flow-shop machines to minimize the makespan.*

Flow shop scheduling is one of the three basic models (open shop, flow shop, job shop) of multi-stage scheduling problems. Flow shop scheduling with minimizing the makespan is usually denoted by $Fm||C_{\max}$, where $m$ is the number of machines. Garey et al. [5] proved that $Fm||C_{\max}$ is strongly NP-hard for $m \geq 3$, and Hall [7] proposed a PTAS algorithm for $Fm||_{\max}$. Particularly, the two-machine flow shop scheduling problem $F2||C_{\max}$ can be solved by Johnson's algorithm in $O(n \log n)$ time [10]. If all the jobs are processed in the same order, then we call this schedule a permutation schedule. It is known that $F2||C_{max}$ or $F3||C_{max}$ has an optimal permutation schedule [2]. The no-wait flow shop scheduling, which is denoted by $Fm|no-wait|C_{\max}$. In $Fm|no-wait|C_{\max}$, once a job has been processed, each stage must be started after its completion of the previous stage without any delay. In the traditional setting, each job has exactly two distinct operations and must be scheduled fulfilling the no-wait restrictions. In other words, each of the two processes of jobs must be scheduled even if it has zero processing time (see, e.g., [3, Section 6.3]). We note that the no-wait 2-FLC problem studied in this paper correspond to this traditional setting of no-wait restriction. Gilmore and Gomory [6] proposed an $O(n \log n)$ algorithm to solve this problem in polynomial time, which is by relating it to a polynomially solvable case of the traveling salesman problem. Researchers also concerned on the problem in which some job may have only one stage, namely, with missing operations. Surprisingly, [22] showed that the problem with missing operations is NP-hard in the strong sense and hence is much harder than that without missing operations. Furthermore, it is observed that any feasible schedule of $F2|no-wait|C_{\max}$ must be a permutation schedule. We refer to the literature [3,8] for more discussion of no-wait scheduling and its applications.

For 2-FLC problem, [15,16] showed that the problem under linear constraints is NP-hard in the strong sense, which sharply differs from the computational complexity of the original $F2||C_{\max}$. In the following, we will show that the no-wait 2-FLC problem is also NP-hard, which is more difficult than its original version. We remark that the hardness result of 2-FLC problem cannot be directly applied to no-wait 2-FLC problem, since the jobs used in the reduction [15,16] do not satisfy no-wait restriction.

To close this section, we state a classic result of $F2|no-wait|C_{\max}$ (see, e.g., [3, Section 6.3]), which will be frequently used in the subsequent analysis. Let $(\sigma(1), \sigma(2), ..., \sigma(n))$ be an arbitrary feasible schedule of $F2|no-wait|C_{\max}$. Then the makespan of this schedule is given by

$$C_{\max} = x_{\sigma(1)} + \sum_{i=2}^{n} \max(y_{\sigma(i-1)}, x_{\sigma(i)}) + y_{\sigma(n)}. \tag{7}$$

## 3.2 Computational Complexity

In this section, we show that the no-wait 2-FLC problem is NP-hard. As mentioned above, we consider the flow-shop scheduling problem without any missing operations. For the original problem without linear constraints, on can use Gilmore and Gomory's algorithm [6] to find an optimal schedule in polynomial time. Our hardness result indicates that the problem under linear constraints is much more difficult. We summarize the result in Theorem 4.

**Theorem 4.** *The no-wait 2-FLC problem is NP-hard, even if each job has strictly positive processing times on both stages.*

## 3.3 Algorithms

In the general case, it can be observed that a straightforward 2-approximation algorithm exists for the no-wait 2-FLC problem, which is based on solving a linear program that minimizes the total processing time of the jobs.

**Theorem 5.** *The no-wait 2-FLC problem has a* 2*-approximation algorithm.*

Next we study a special case where the no-wait 2-FLC problem can be solved in polynomial time, in which the number of constraints $k$ is fixed. The key is to prove a similar result as the SSLC problem, which shows that the no-wait 2-FLC problem has an optimal solution with a fixed number of nonzero processing time jobs if $k$ is fixed. We remark that the case of no-wait 2-FLC problem is more complicated than Lemma 2, since the objective function (7) is not linear in general even when the optimal schedule $\sigma$ is known.

**Lemma 2.** *The no-wait 2-FLC problem has an optimal solution, in which each machine has at most $k$ jobs with nonzero processing time.*

Based on Lemma 2, we propose an enumeration algorithm for 2-FLC problem with a fixed number of constraints $k$. We summarize it in Algorithm 2 and Theorem 6.

**Theorem 6.** *Algorithm 2 returns an optimal solution to the no-wait 2FLC problem and has time complexity $O(n^{2k}L)$, where the parameter $L$ is the input length of (8).*

**Algorithm 2.** Enumeration algorithm for 2-FLC problem with fixed $k$

```
1: for each subset $J'$ of $J$ with $k$ jobs do
2:     for each possible permutation of the jobs in $J'$ do
3:         Let $(\sigma(1), ..., \sigma(k))$ be the permutation, i.e., the schedule of these $k$ jobs. Solve
           the following LP while setting $y_i = 0$ for $i \notin J'$.
```

$$\min \quad x_{\sigma(1)} + \frac{1}{2}\sum_{i=2}^{n}(y_{\sigma(i-1)} + x_{\sigma(i)} + z_i^+ + z_i^-) + y_{\sigma(n)} \tag{8a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} a_{ji}x_i + c_{ji}y_i \geq b_j \qquad \forall\, j = 1, \ldots, k \tag{8b}$$

$$x_{\sigma(i)} + z_i^+ = y_{\sigma(i-1)} + z_i^- \qquad \forall\, i = 2, \ldots, n \tag{8c}$$

$$z_i^+, z_i^-, x_i, y_i \geq 0 \qquad \forall\, i = 1, \ldots, n,$$

```
4:         if (8) is feasible then
5:             Let the processing times of jobs be the optimal solution to (8), and record
               the schedule and the makespan.
6: return the schedule with the smallest makespan among all these iterations and its
   corresponding processing times.
```

## 4 Conclusions

In this work, we investigate the computational complexity of two different scheduling problems under linear constraints. We show that the problems with linear constraints are NP-hard, while the original versions of both problems are polynomially solvable. Additionally, we propose polynomial time exact or approximation algorithms for various cases of the problems. One potential research direction is to develop improved approximation algorithms for the general cases of these problems, or to show the possibility of inapproximability. Moreover, it is interesting to explore other types of objectives (e.g., lateness or tardiness) or machine restrictions (e.g., release date, due dates or job unavailability) for the machine scheduling problems under linear constraints.

## References

1. Bruno, J.L., Coffman, E.G., Jr., Sethi, R.: Scheduling independent tasks to reduce mean finishing time. Commun. ACM **17**(7), 382–387 (1974)
2. Conway, R.W., Maxwell, W.L., Miller, L.W.: Theory of Scheduling. Reading (1967)
3. Emmons, H., Vairaktarakis, G.: Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-5152-5
4. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)

5. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Math. Oper. Res. **1**, 117–129 (1976)
6. Gilmore, C., Gomory, R.E.: Sequencing a one state-variable machine: a solvable case of the travelling salesman problem. Oper. Res. **12**, 655–679 (1964)
7. Hall, L.A.: Approximability of flow shop scheduling. Math. Program. **82**, 175–190 (1998)
8. Hall, N.G., Sriskandarajah, C.: A survey of machine scheduling problems with blocking and no-wait in process. Oper. Res. **44**(3), 510–525 (1996)
9. Jansen, K., Lassota, A., Maack, M., Pikies, T.: Total completion time minimization for scheduling with incompatibility cliques. In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 31, no. 1, pp. 192–200 (2021)
10. Johnson, S.M.: Optimal two- and three-stage production schedules with setup times included. Naval Res. Logist. Q. **1**, 61–68 (1954)
11. Knop, D., Koutecký, M.: Scheduling meets n-fold integer programming. J. Sched. **21**(5), 493–503 (2018)
12. Lawler, J.L., Johnson, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: The complexity of machine scheduling problems. Ann. Discrete Math. **1**, 343–362 (1977)
13. Luenberger, D.G., Ye, Y.: Linear and Nonlinear Programming, 4th edn. Springer, Cham (2016)
14. Nip, K., Shi, T., Wang, Z.: Some graph optimization problems with weights satisfying linear constraints. J. Comb. Optim. **43**, 200–225 (2022)
15. Nip, K., Wang, Z.: Two-machine flow shop scheduling problem under linear constraints. In: Li, Y., Cardei, M., Huang, Y. (eds.) COCOA 2019. LNCS, vol. 11949, pp. 400–411. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36412-0_32
16. Nip, K., Wang, Z.: A complexity analysis and algorithms for two-machine shop scheduling problems under linear constraints. J. Sched. (2021)
17. Nip, K., Wang, Z., Shi, T.: Some graph optimization problems with weights satisfying linear constraints. In: Li, Y., Cardei, M., Huang, Y. (eds.) COCOA 2019. LNCS, vol. 11949, pp. 412–424. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36412-0_33
18. Nip, K., Wang, Z., Wang, Z.: Scheduling under linear constraints. Eur. J. Oper. Res. **253**(2), 290–297 (2016)
19. Nip, K., Wang, Z., Wang, Z.: Knapsack with variable weights satisfying linear constraints. J. Global Optim. **69**(3), 713–725 (2017). https://doi.org/10.1007/s10898-017-0540-y
20. Ogryczak, W., Śliwiński, T.: On solving linear programs with the ordered weighted averaging objective. Eur. J. Oper. Res. **148**(1), 80–91 (2003)
21. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems. Springer, New York (2016)
22. Sahni, S., Cho, Y.: Complexity of scheduling shops with no wait in process. Math. Oper. Res. **4**(4), 448–457 (1979)
23. Smith, W.E.: Various optimizers for single-stage production. Naval Res. Logist. Q. **3**, 59–66 (1956)
24. Wang, Z., Nip, K.: Bin packing under linear constraints. J. Comb. Optim. **34**(4), 1198–1209 (2017). https://doi.org/10.1007/s10878-017-0140-2
25. Yager, R.R.: On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Trans. Syst. Man Cybern. **18**(1), 183–190 (1988)

26. Yager, R.R.: Constrained OWA aggregation. Fuzzy Sets Syst. **81**(1), 89–101 (1996)
27. Zhang, S., Nip, K., Wang, Z.: Related machine scheduling with machine speeds satisfying linear constraints. In: Kim, D., Uma, R.N., Zelikovsky, A. (eds.) COCOA 2018. LNCS, vol. 11346, pp. 314–328. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04651-4_21
28. Zhang, S., Nip, K., Wang, Z.: Related machine scheduling with machine speeds satisfying linear constraints. J. Comb. Optim. **44**(3), 1724–1740 (2022)