Int. J. Appl. Math. Comput. Sci., 2024, Vol. 34, No. 2, 253–261
DOI: 10.61822/amcs-2024-0018

ONLINE AND SEMI–ONLINE SCHEDULING ON TWO HIERARCHICAL MACHINES WITH A COMMON DUE DATE TO MAXIMIZE THE TOTAL EARLY WORK

MAN XIAO [a], XIAOQIAO LIU [a], WEIDONG LI [a], XIN CHEN [b,*], MALGORZATA STERNA [c], JACEK BLAZEWICZ [c,d]

[a]School of Mathematics and Statistics
Yunnan University
Wujiaying, Kunming 650504, China
e-mail: man1205@163.com, 1161407532@qq.com, weidongmath@126.com

[b]School of Electronics and Information Engineering
Liaoning University of Technology
Shiying 169, Jinzhou 121001, China
e-mail: chenxin.lut@hotmail.com

[c]Institute of Computing Science
Poznan University of Technology
ul. Piotrowo 2, 60-965 Poznań, Poland
e-mail: {malgorzata.sterna, jblazewicz}@cs.put.poznan.pl

[d]European Centre for Bioinformatics and Genomics
Polish Academy of Sciences
ul. Piotrowo 2, 60-965 Poznań, Poland

In this study, we investigate several online and semi-online scheduling problems related to two hierarchical machines with a common due date to maximize the total early work. For the pure online case, we design an optimal online algorithm with a competitive ratio of $\sqrt{2}$. Additionally, for the cases when the largest processing time is known, we give optimal algorithms with a competitive ratio of $6/5$ if the largest job is a lower-hierarchy one, and of $\sqrt{5}-1$ if the largest job is a higher-hierarchy one.

Keywords: online and semi-online, early work, hierarchical scheduling, competitive ratio.

1. Introduction

Scheduling with a due date (Gordon *et al.*, 2002) or a due window (Janiak *et al.*, 2013) characterizes the time accuracy in a real production environment. Among them, early work (Sterna, 2021) is one of the typical scheduling criteria related to the due date. Early work scheduling on parallel machines with a common due date (Chen *et al.*, 2016) involves executing n jobs on m parallel identical machines in a non-overlapping and non-preemptive manner, to maximize the total early work of all the jobs. The early work of a job denotes its part executed before the common due date.

From the optimization point of view, early work maximization is correlated to late work minimization, since the late work of a job denotes its part executed after this due date. Both measures have been widely investigated over several years (Sterna, 2011; 2021) from a theoretical perspective as well as within various practical applications, such as in control systems when collecting data from sensors, in agriculture in the process of harvesting crops, in manufacturing systems when planning technological processes, and in software

*Corresponding author

engineering in the process of software testing. Late and early work are obviously equivalent when the optimal solutions should be determined, but the studies on offline approximation and online algorithms have been done mostly for early work due to the properties of the two criteria (late work in an optimal schedule could be zero so that it cannot be used as a denominator in the case of ratio analysis).

For an offline maximization problem, a ρ-approximation algorithm A is a polynomial time algorithm that always produces a feasible solution (for any instance I with a minimum ρ) satisfying $\frac{C^{\text{OPT}}(I)}{C^{A}(I)} \leq \rho$, in which $C^{\text{OPT}}(I)$(C^{OPT} for short) denotes the optimal criterion value, and $C^{A}(I)$ (C^{A} for short) denotes the output value by A.

In the case of online scheduling problems where the set of jobs is unknown in advance and jobs arrive one by one, the lack of knowledge on the problem instance limits the optimization process. However, it is possible to construct online algorithms, whose efficiency is evaluated in a similar way to the above-mentioned offline approximation algorithms.

The performance of an online algorithm is measured by its competitive ratio. Similar with the approximation ratio, the *competitive ratio* of an online algorithm A for a maximization problem is defined as the minimum r such that $\frac{C^{\text{OPT}}}{C^{A}} \leq r$ holds for any problem instance. Then we claim that the online problem considered has an *upper bound* of r. On the other hand, proving that no online algorithm has a competitive ratio less than δ means that δ is a *lower bound* of this problem. In consequence, an online algorithm is called *optimal* (or the *best possible*) if its competitive ratio r equals the problem lower bound δ, i.e., $r = \delta$. In such a case, we say that this problem has a *tight bound*.

Previous studies on early work maximization for parallel identical machines and a common due date focus mainly on offline scheduling. The non-preemptive problem is weakly NP-hard for the fixed number of machines, while for an arbitrary number of machines it becomes strongly NP-hard (Chen *et al.*, 2016). For the two-machine model ($m = 2$), Sterna and Czerniachowska (2017) designed a PTAS (polynomial time approximation scheme) based on structuring the problem input. Chen *et al.* (2020b) proved that the classical LPT (longest processing time first) heuristic has an approximation ratio of $\frac{10}{9}$, and then Jiang *et al.* (2021) demonstrated that the tight bound of LPT is exactly $\frac{12}{11}$. For the more general case where the number of machine m is fixed ($m \geq 2$), Chen *et al.* (2020a) proposed an FPTAS (fully polynomial time approximation scheme) based on a dynamic programming approach. Later, for the cases with an arbitrary number of machines, Györgyi and Kis (2020) introduced a PTAS, while Li (2024) proposed an EPTAS (efficient polynomial time approximation scheme). Recently, Chen *et al.* (2022) improved their previous dynamic programming and suggested two new FPTASs for the m-machine case. For the model of the unrelated machines, Wang *et al.* (2020) proposed two meta-heuristic algorithms. Recently, Liu *et al.* (2023) introduced a mathematical model and two dedicated exact approaches, based on the branching and bounding strategy and on enumerating combined with a dynamic programming algorithm.

The total early work maximization was introduced into online environment by Chen *et al.* (2016), who designed an optimal online algorithm with a competitive ratio of $\sqrt{5} - 1$ for two identical machines and a common due date case. (Note that they used the context of late work minimization at that time.) Then, this group (Chen *et al.*, 2021) continued this topic on several semi-online models, where some partial knowledge of the problem instances is available in advance. For the cases when the total processing time or the optimal criterion value is known, they designed an optimal online algorithm with a competitive ratio of $\frac{6}{5}$. For the cases when the maximal job processing time is known, they obtained a lower bound of 1.1231 and an upper bound of 1.1375, respectively.

In the scheduling field, the hierarchical constraint (Bar-Noy *et al.*, 2001) means that jobs may have different hierarchies, and some of the machines can process all types of jobs, while others are dedicated to high-hierarchy jobs only. For example, in a bank service, some windows are open for all the customers, but some of them are for VIPs (very important person) only. This constraint has been introduced into online scheduling independently by Jiang *et al.* (2006) and Park *et al.* (2006), when both of the two groups considered makespan minimization on identical machines. Jiang (2008) later generalized his results (Jiang *et al.*, 2006). Luo and Xu (2015) studied online and semi-online hierarchical scheduling on two identical machines with the goal of maximizing the minimum machine load, and proposed several optimal algorithms. Recently, Akaria and Epstein (2022) revisited the semi-online scheduling problem on two hierarchical machines, in which bounded migration is allowed, and showed the best possible result for the considered problem.

Hierarchical scheduling was combined with early work maximization in our previous work (Xiao *et al.*, 2021) for the first time, when we studied three semi-online models assuming that jobs' total size is provided in advance. Specifically, when the total size of the lower or higher hierarchy is known, the tight bound was proven to be $\sqrt{5} - 1$. Additionally, if both pieces of information are known, the tight bound was shown to be $\frac{6}{5}$. Very recently, Xiao *et al.* (2023) further considered these models on two uniform machines, where the speeds of the two machines are s and 1, respectively. They designed an optimal online

Table 1. Relevant results.

Model		Tight bound	Reference
Pure online		$\sqrt{2}$	This paper
Semi-online	T_1	$\sqrt{5}-1$	Xiao *et al.*, 2021
	T_2	$\sqrt{5}-1$	Xiao *et al.*, 2021
	$T_1\&T_2$	$\frac{6}{5}$	Xiao *et al.*, 2021
	T	$\frac{4}{3}$	Xiao *et al.*, 2023
	$p_{\max}$	$\sqrt{2}$	This paper
	$p_{\max,1}$	$\frac{6}{5}$	This paper
	$p_{\max,2}$	$\sqrt{5}-1$	This paper

algorithm with a competitive ratio of $\min\{1+s, \frac{2+2s}{1+2s}\}$ for the case when the total processing time of all jobs is known. Then, two optimal online algorithms were given for the cases when the total processing time of low- and high-hierarchy jobs is known with competitive ratios of $\min\{1+s, \frac{\sqrt{9s^2+10+1}-s-1}{2s}\}$ and $\min\{\sqrt{s+1}, \sqrt{s^2+2s+2}-s\}$, respectively. When both mentioned pieces of information are provided, they proved a lower bound $\frac{2s+2}{s+2}$ when $s \le \frac{2}{3}$, and designed an optimal online algorithm with a competitive ratio of $\frac{3s+3}{3s+2}$ for $s > \frac{2}{3}$.

Motivated by the previous works mentioned above, we consider several online and semi-online models for early work maximization related to two identical machines with the hierarchical constraint and a common due date, and propose efficient approaches for them. In this paper, our primary contributions are outlined as follows:

- First, we consider the pure online version without any information on the job sequence, and design an optimal algorithm with a competitive ratio of $\sqrt{2}$.
- Then, for the cases when the maximal job processing time is known, we design optimal semi-online algorithms with competitive ratios of $\frac{6}{5}$ and $\sqrt{5}-1$, depending on whether the largest job is of a lower hierarchy or a higher hierarchy, respectively.

For a clear comparison of the results in this paper, we summarize the relevant ones on hierarchical early work maximization problems with two identical machines in Table 1.

The rest of this paper is organized as follows. In Section 2, we give the formal definition for the problem considered. Then, the pure online case is analysed in Section 3, followed by the analysis of the case when the largest processing time is known in Section 4. Finally, we present conclusions and possible directions for future work in Section 5.

2. Problem definition

The problem studied in this paper can be defined as follows.

There are two identical machines $M = \{M_1,\ M_2\}$, and a set of jobs $J = \{J_1, J_2, \dots, J_n\}$ which come to the system one by one. We say that these jobs come "over a list", contrary to the online problems where jobs are released at a given time (i.e., "over time"). Once a job arrives, it has to be scheduled immediately and irrevocably on one of the machines. The j-th job J_j is described by two parameters (p_j, g_j), where p_j is the processing time and $g_j \in \{1,\ 2\}$ is the hierarchy of this job. The hierarchical constraint means that the machines have different capabilities when processing jobs, i.e., M_1 is a general machine which is available for all the jobs, while M_2 is a specific one and available only for the high-hierarchy jobs (such as VIP service in a bank system). Hence, M_1 can process all the jobs with $g_j \in \{1,2\}$, whereas M_2 can process only jobs of hierarchy 2 ($g_j = 2$). Moreover, we assume that all the jobs share a common due date $d > 0$ ($d_j = d$), and $p_j \le d$ for $j = 1,\ 2,\ \dots,\ n$.

The goal is to schedule these jobs on the machines without preemption, to maximize the total early work of all jobs. The early work of job J_j, denoted as X_j, is the part of J_j executed before the common due date d, i.e., $X_j = \min\{p_j, \max\{0, d-(C_j-p_j)\}\}$, where C_j is the completion time of J_j. More precisely, if J_j is completed before d ($C_j \le d$), this job is called totally early and we have $X_j = p_j$. If J_j starts before d, ($C_j - p_j < d$), but finishes its execution after d ($C_j > d$), we say that this job is partially early (or partially late) and $X_j = d-(C_j-p_j)$. Finally, if J_j starts its execution after d ($C_j - p_j \ge d$), this job is totally late and we have no profit in this case, i.e., $X_j = 0$.

Using the three-field notation, which is commonly applied in the scheduling domain (Graham *et al.*, 1979), the pure online model of the problem considered can be denoted as $P2|GoS, online, d_j = d|\max(X)$, in which GoS (Grade of Service) stands for the hierarchical constraint described above.

We can see that a schedule of J on M can be considered a partition $(S_1,\ S_2)$ of all jobs to the two machines, such that $S_1 \cup S_2 = J$ and $S_1 \cap S_2 = \varnothing$. We define the load of machine M_i ($i \in \{1,2\}$) by L_i, i.e., $L_i = \sum_{J_j \in S_i} p_j$. Then the early work of jobs assigned to M_i is equal to $\min\{L_i, d\}$. Thus, the aim of the scheduling is to find a partition such that $X = \sum_{j=1}^{n} X_j = \sum_{i=1}^{2} \min\{L_i, d\}$ is maximized.

Let L_i^j be the load of M_i after job J_j is assigned to one of the machines ($i \in \{1,2\}$ and $1 \le j \le n$), so that we have $L_i = L_i^n$ for each machine. Moreover, let T_k ($k \in \{1,2\}$) be the total processing time of the jobs with hierarchy k, and let T be the total processing

Table 2. Some important symbols.

L_i^j	Load of M_i after job J_j is assigned to one of the machines, and $L_i = L_i^n$.
T_k	Total processing time of the jobs with hierarchy k, and $T = T_1 + T_2$.
$p_{\max,k}$	Largest job processing time with hierarchy k.
X^{OPT}	Optimal criterion value.
X^A	Criterion value obtained by an algorithm A.

time of all jobs, i.e., $T = T_1 + T_2$. For any instance of problem $P2|GoS, online, d_j = d|\max(X)$, let X^{OPT} be its optimal criterion value, and X^A be the criterion value obtained by an algorithm A. For convenience, we also list some important symbols in Table 2.

Based on the definitions, we have the following result.

Lemma 1. *In the problem $P2|GoS, online, d_j = d|\max(X)$, the optimal criterion value X^{OPT} satisfies*

$$X^{OPT} \le \min\{T, 2d\} \le d + \frac{T}{2}.$$

In the pure online case, i.e., $P2|GoS, online, d_j = d|\max(X)$, no additional knowledge on problem instances is given (cf. Section 3). However, if some information (not all) is given in advance, i.e., in semi-online cases, we can utilize this knowledge to achieve more efficient procedures. We denote by $P2|GoS, online, d_j = d, \Delta|\max(X)$ the semi-online problem if some information on the maximum processing time is known, where $\Delta \in \{p_{\max}, p_{\max,1}, p_{\max,2}\}$ stands for the maximum processing time ($p_{\max}$), along with its hierarchy $k \in \{1, 2\}$ ($p_{\max,1}$ and $p_{\max,2}$) (cf. Section 4).

3. Pure online case

In this section, we study the online problem $P2|GoS, online, d_j = d|\max(X)$, where no information on problem instances is available *a priori*. We prove a lower bound of $\sqrt{2}$ for this problem, and provide an optimal online algorithm solving it with a competitive ratio of $\sqrt{2}$. It is worth mentioning that this model has not been studied before, even in our previous research (Xiao *et al.*, 2021; 2023).

Theorem 1. *Any online algorithm A for the problem $P2|GoS, online, d_j = d|\max(X)$ has a competitive ratio of at least $\sqrt{2}$.*

Proof. Let $d = 1$, and the first job in the job sequence be $J_1 = (\sqrt{2} - 1, 2)$.

Case 1: If J_1 is assigned to M_1, then the last job is $J_2 = (1, 1)$, which implies that $X^A = 1$. However, we have $X^{\text{OPT}} = \sqrt{2}$, by assigning J_2 to M_1, and J_1 to M_2.

Case 2: If J_1 is assigned to M_2, the second job $J_2 = (1, 2)$ arrives.

Case 2.1: If J_2 is assigned to M_2, no more jobs arrive, implying that $X^A = 1$. But we have $X^{\text{OPT}} = \sqrt{2}$ again, by assigning J_1 and J_2 to different machines.

Case 2.2: If J_2 is assigned to M_1, the last job is $J_3 = (1, 1)$ and we get $X^A = \sqrt{2}$. However, we have $X^{\text{OPT}} = 2$, which is obtained by assigning J_3 to M_1, and $\{J_1, J_2\}$ to M_2.

Therefore, in any case, we have $\frac{X^{\text{OPT}}}{X^A} \ge \sqrt{2}$, and the theorem holds. ■

Now we propose an optimal online algorithm solving the pure online scheduling problem considered, presented as Algorithm 1. The primary concept of this algorithm is that large jobs are not assigned to M_2 (the machine dedicated for high-hierarchy jobs only) unless the current load of M_2 is relatively small, which helps to keep a threshold of the load on this machine.

Theorem 2. *The competitive ratio of Algorithm 1 is $\sqrt{2}$.*

Proof. Based on Lemma 1, if $\min\{L_1, L_2\} \ge d$, we have $X^A = 2d \ge X^{\text{OPT}}$. If $\max\{L_1, L_2\} \le d$, we have $X^A = T \ge X^{\text{OPT}}$. This implies that we only need to consider the case when $\min\{L_1, L_2\} < d < \max\{L_1, L_2\}$, implying that

$$X^A = d + \min\{L_1, L_2\}.$$

Subsequently, we distinguish the following two cases:

Case 1: $\max\{L_1, L_2\} = L_1 > d$. In this case, we have $X^A = d + L_2$. If there is no job of hierarchy 2 assigned to M_1, we have $L_1 = T_1 > d$ and $L_2 = T_2 < d$, implying that Algorithm 1 reaches the optimal solution. Otherwise, let $J_l = (p_l, 2)$ be the last job of hierarchy 2 assigned to M_1. By the choice of Algorithm 1, we have $L_2 \ge L_2^{l-1} > (\sqrt{2} - 1)d$. By Lemma 1, we have

$$\frac{X^{\text{OPT}}}{X^A} \le \frac{2d}{d + L_2} \le \frac{2d}{d + (\sqrt{2} - 1)d} = \sqrt{2}.$$

Case 2: $\max\{L_1, L_2\} = L_2 > d$. In this case, we have $X^A = d + L_1$. Let $L_{1,2}$ be the total processing time of jobs of hierarchy 2 assigned to M_1, and $J_l = (p_l, 2)$ be the last job assigned to M_2.

If J_l is assigned to M_2 in Line 7 of the algorithm, we have $L_2 = L_2^l = L_2^{l-1} + p_l \le d$, contradicting the assumption $L_2 > d$.

If J_l is assigned to M_2 in Line 10, we have $L_2 = L_2^l = L_2^{l-1} + p_l > d$ and $L_2^{l-1} \le (\sqrt{2} - 1)d$. Therefore, $T_2 = L_2^{l-1} + p_l + L_{1,2} \le (\sqrt{2} - 1)d + p_l + L_{1,2} \le$

Algorithm 1. A1.

```
1: Initially, let $L_2^0 = 0$ and $j = 1$;
2: When a new job $J_j = (p_j, g_j)$ arrives,
3: if $g_j = 1$ then
4:    Assign $J_j$ to $M_1$, and set $L_2^j = L_2^{j-1}$.
5: else
6:    if $L_2^{j-1} + p_j \le d$ then
7:       Assign $J_j$ to $M_2$, and set $L_2^j = L_2^{j-1} + p_j$.
8:    else
9:       if $L_2^{j-1} \le (\sqrt{2} - 1)d$ then
10:         Assign $J_j$ to $M_2$, and set $L_2^j = L_2^{j-1} + p_j$.
11:      else
12:         Assign $J_j$ to $M_1$, and set $L_2^j = L_2^{j-1}$.
13:      end if
14:   end if
15: end if
```

$\sqrt{2}d + L_{1,2}$, where the last inequality follows from the assumption $p_l \le d$. Based on Lemma 1, we have

$$\frac{X^{\text{OPT}}}{X^A} \le \frac{T}{d + L_1} = \frac{T_1 + T_2}{d + T_1 + L_{1,2}} \le \frac{T_1 + \sqrt{2}d + L_{1,2}}{d + T_1 + L_{1,2}} \le \sqrt{2}.$$

■

4. Semi-online case with the largest processing time known

In this section, we consider a series of models when some information on the largest job processing time is known in advance, which we denote as $P2|GoS, online, d_j = d, \Delta|\max(X)$, and $\Delta \in \{p_{\max}, p_{\max,1}, p_{\max,2}\}$. If $\Delta = p_{\max}$, this means that we know the information on the largest processing time of the input jobs. Unfortunately, just this information is not helpful, since by applying the same instance as in the proof of Theorem 1 we can get a lower bound of $\sqrt{2}$ for the problem $P2|GoS, online, d_j = d, p_{\max}|\max(X)$, even we know that the largest job processing time is 1. This means that we cannot obtain a better semi-online algorithm with the knowledge of $p_{\max}$, compared with the pure online case.

However, the bounds could be reduced if we knew more about the largest processing time of the jobs. We define $\Delta = p_{\max,k}$ if we know $p_{\max}$, and also know that the hierarchy of the job with processing time $p_{\max}$ is k ($k \in \{1, 2\}$). For $\Delta = p_{\max,1}$, i.e., for the problem $P2|GoS, online, d_j = d, p_{\max,1}|\max(X)$, we prove a tight bound of $\frac{6}{5}$ (cf. Section 4.1). In contrast, if the hierarchy of the largest job is 2 ($\Delta = p_{\max,2}$), i.e., for the problem $P2|GoS, online, d_j = d, p_{\max,2}|\max(X)$, a tight bound is proven to be $\sqrt{5} - 1$ (cf. Section 4.2).

4.1. Largest job with a low hierarchy. In this subsection, we focus on the semi-online case when the largest job has hierarchy 1, allowing its processing on machine M_1 only ($\Delta = p_{\max,1}$), i.e., for any job $J_j = (p_j, g_j)$, we have

$$p_j \le p_{\max,1} \le d.$$

We prove the lower bound of this problem in Theorem 3, and then propose an optimal semi-online algorithm.

Theorem 3. *Any online algorithm A for the problem $P2|GoS, online, d_j = d, p_{\max,1}|\max(X)$ has a competitive ratio of at least $\frac{6}{5}$.*

Proof. Let $d = 1$ and $p_{\max,1} = \frac{2}{3}$. The first two jobs are $J_1 = (\frac{2}{3}, 1)$ and $J_2 = (\frac{1}{3}, 2)$. Job J_1 can only be assigned to M_1.

Case 1: If J_2 is assigned to M_1, the last job $J_3 = (\frac{2}{3}, 1)$ arrives, implying that $X^A = 1$. However, we have $X^{\text{OPT}} = \frac{4}{3}$ by assigning $\{J_1, J_3\}$ to M_1, and J_2 to M_2.

Case 2: If J_2 is assigned to M_2, the next job $J_3 = (\frac{1}{3}, 2)$ arrives. Subsequently, we distinguish the following two subcases:

Case 2.1: If J_3 is assigned to M_1, the last job $J_4 = (\frac{2}{3}, 1)$ arrives, implying that $X^A = \frac{4}{3}$. But we can get $X^{\text{OPT}} = \frac{5}{3}$ by assigning $\{J_1, J_4\}$ to M_1, and $\{J_2, J_3\}$ to M_2.

Case 2.2: If J_3 is assigned to M_2, the next job $J_4 = (\frac{2}{3}, 2)$ arrives.

If J_4 is assigned to M_1, the last job $J_5 = (\frac{2}{3}, 1)$ arrives, implying that $X^A = \frac{5}{3}$. On the other hand, by the assignment of $\{J_1, J_5\}$ to M_1 and $\{J_2, J_3, J_4\}$ to M_2, we get $X^{\text{OPT}} = 2$.

If J_4 is assigned to M_2, then no more jobs arrive, implying that $X^A = \frac{5}{3}$. In this case, an optimal solution can be get by assigning $\{J_1, J_3\}$ to M_1 and $\{J_2, J_4\}$ to M_2, with $X^{\text{OPT}} = 2$.

Thus, $\frac{X^{\text{OPT}}}{X^A} \ge \frac{6}{5}$ in any case. ■

Now we propose an optimal semi-online algorithm, i.e., Algorithm 2, to solve the problem $P2|GoS, online, d_j = d, p_{\max,1}|\max(X)$. The main idea of this algorithm is as follows: we keep assigning jobs with hierarchy 2 to machine M_2, until its load first exceeds a threshold $\frac{2d}{3}$. The competitive ratio of Algorithm 2 is proven in Theorem 4.

Theorem 4. *The competitive ratio of Algorithm 2 is $\frac{6}{5}$.*

Proof. As before, if $\min\{L_1, L_2\} \ge d$ or $\max\{L_1, L_2\} \le d$, Algorithm 2 reaches the optimal solution. We only need to consider the case when $\min\{L_1, L_2\} < d < \max\{L_1, L_2\}$, which implies that

$$X^A = d + \min\{L_1, L_2\}.$$

Algorithm 2. A2.

```
1:  Initially, let $L_2^0 = 0$ and $j = 1$.
2:  When a new job $J_j = (p_j, g_j)$ arrives,
3:  if $g_j = 1$ then
4:      Assign $J_j$ to $M_1$, and set $L_2^j = L_2^{j-1}$.
5:  else
6:      if $L_2^{j-1} < \frac{2d}{3}$ then
7:          Assign $J_j$ to $M_2$, and set $L_2^j = L_2^{j-1} + p_j$.
8:      else
9:          Assign $J_j$ to $M_1$, and set $L_2^j = L_2^{j-1}$.
10:     end if
11: end if
```

We distinguish the following two cases.

Case 1: $\max\{L_1, L_2\} = L_1 > d$. In this case, we have $X^A = d + L_2$. If there is no job of hierarchy 2 assigned to M_1, we have $L_1 = T_1 > d$ and $L_2 = T_2 < d$, which implies that Algorithm 2 reaches the optimal solution. Otherwise, let $J_l = (p_l, 2)$ be the last job of hierarchy 2 assigned to M_1. By the choice of Algorithm 2, we have $L_2 \geq L_2^{l-1} \geq \frac{2d}{3}$. Based on Lemma 1, we have

$$\frac{X^{\text{OPT}}}{X^A} \leq \frac{2d}{d + L_2} \leq \frac{2d}{d + \frac{2d}{3}} = \frac{6}{5}.$$

Case 2: $\max\{L_1, L_2\} = L_2 > d$. In this case, we have $X^A = d + L_1$. If $p_{\max,1} \geq \frac{2d}{3}$, we have $L_1 \geq p_{\max,1} \geq \frac{2d}{3}$. Therefore,

$$\frac{X^{\text{OPT}}}{X^A} \leq \frac{2d}{d + L_1} \leq \frac{2d}{d + \frac{2d}{3}} = \frac{6}{5}.$$

If $p_{\max,1} < \frac{2d}{3}$, let $J_l = (p_l, 2)$ be the last job assigned to M_2. Assume that $p_l = \alpha d \leq p_{\max,1}$, which implies $\alpha < \frac{2}{3}$. By the choice of Algorithm 2, we have $L_2^{l-1} < \frac{2d}{3}$ and $L_2 = L_2^l = L_2^{l-1} + p_l < \frac{2d}{3} + \alpha d$. Based on Lemma 1, we have

$$\begin{aligned}\frac{X^{\text{OPT}}}{X^A} &\leq \frac{T}{d + L_1} = \frac{L_1 + L_2}{d + L_1} = 1 + \frac{L_2 - d}{L_1 + d} \\ &< 1 + \frac{\frac{2d}{3} + \alpha d - d}{L_1 + d}.\end{aligned}$$

Since $L_1 \geq p_{\max,1} \geq p_l = \alpha d$, we have

$$\begin{aligned}\frac{X^{\text{OPT}}}{X^A} &\leq 1 + \frac{\frac{2d}{3} + \alpha d - d}{L_1 + d} \leq 1 + \frac{(\alpha - \frac{1}{3})d}{(1 + \alpha)d} \\ &= 1 + \frac{3\alpha - 1}{3 + 3\alpha} = 2 - \frac{4}{3 + 3\alpha} < \frac{6}{5},\end{aligned}$$

where the last inequality follows from the fact that $\alpha < \frac{2}{3}$. ■

4.2. Largest job with a high hierarchy. In this subsection, we focus on the case when the largest job has hierarchy 2, allowing its processing on both machines ($\Delta = p_{\max,2}$), i.e., for any job $J_j = (p_j, g_j)$, we have

$$p_j \leq p_{\max,2} \leq d.$$

As for the previous case, we prove the lower bound of this problem in Theorem 5, and propose an optimal semi-online algorithm, i.e., Algorithm 3.

Theorem 5. *Any online algorithm A for the problem $P2|GoS, online, d_j = d, p_{\max,2}|\max(X)$ has a competitive ratio at least $\sqrt{5} - 1$.*

Proof. Let $d = 1$ and $p_{\max,2} = \frac{\sqrt{5}-1}{2}$. The first job is $J_1 = (\frac{\sqrt{5}-1}{2}, 2)$.

Case 1: If J_1 is assigned to M_1, the last two jobs $J_2 = (\frac{1}{2}, 1)$ and $J_3 = (\frac{1}{2}, 1)$ arrive, which implies that $X^A = 1$. But we can get $X^{\text{OPT}} = \frac{\sqrt{5}+1}{2}$ by assigning $\{J_2, J_3\}$ to M_1, and J_1 to M_2.

Case 2: If J_1 is assigned to M_2, the next job $J_2 = (\frac{\sqrt{5}-1}{2}, 2)$ arrives.

Case 2.1: If J_2 is assigned to M_1, the last two jobs $J_3 = (\frac{1}{2}, 1)$ and $J_4 = (\frac{1}{2}, 1)$ arrive, which implies that $X^A = 1 + \frac{\sqrt{5}-1}{2} = \frac{\sqrt{5}+1}{2}$. However, $X^{\text{OPT}} = 2$ by the assignment of $\{J_3, J_4\}$ to M_1, and $\{J_1, J_2\}$ to M_2.

Case 2.2: If J_2 is assigned to M_2, no more jobs arrive, which implies that $X^A = 1$. In this situation, $X^{\text{OPT}} = \sqrt{5} - 1$ by assigning J_1 and J_2 to different machines.

Thus, $\frac{X^{\text{OPT}}}{X^A} \geq \sqrt{5} - 1$ in any case. ■

The main idea of Algorithm 3 is to preserve machine M_2 for the first largest job in the input sequence until it appears. We use $n_2 = 0$ to denote that the first largest job has not appeared, and n_2 is set to be 1 when this job comes to the system. The competitive ratio of this approach is proven in Theorem 6.

Theorem 6. *Algorithm 3 achieves a competitive ratio of $\sqrt{5} - 1$.*

Proof. As before, whenever $\min\{L_1, L_2\} \geq d$ or $\max\{L_1, L_2\} \leq d$, Algorithm 3 reaches the optimal solution. We only need to consider the case when $\min\{L_1, L_2\} < d < \max\{L_1, L_2\}$, implying that

$$X^A = d + \min\{L_1, L_2\}.$$

Subsequently, we distinguish the following two cases.

Case 1: $\max\{L_1, L_2\} = L_1 > d$. In this case, we have $X^A = d + L_2$. If there is no job of hierarchy 2 assigned to M_1, we have $L_1 = T_1 > d$ and $L_2 = T_2 < d$, implying

Algorithm 3. A3.

```
1:  Initially, let $L_2^0 = 0$, $j = 1$ and $n_2 = 0$.
2:  When a new job $J_j = (p_j, g_j)$ arrives,
3:  if $g_j = 1$ then
4:      Assign $J_j$ to $M_1$, and set $L_2^j = L_2^{j-1}$.
5:  else
6:      if $n_2 = 0$ && $p_j \neq p_{\max,2}$ then
7:          if $L_2^{j-1} + p_{\max,2} + p_j \leq (\sqrt{5}-1)d$ then
8:              Assign $J_j$ to $M_2$, and set $L_2^j = L_2^{j-1} + p_j$.
9:          else
10:             Assign $J_j$ to $M_1$, and set $L_2^j = L_2^{j-1}$.
11:         end if
12:     else
13:         if $n_2 = 0$ && $p_j = p_{\max,2}$ then
14:             Set $n_2 = 1$, assign $J_j$ to $M_2$, and set $L_2^j = L_2^{j-1} + p_j$.
15:         else
16:             if $L_2^{j-1} + p_j \leq (\sqrt{5}-1)d$ then
17:                 Assign $J_j$ to $M_2$, and set $L_2^j = L_2^{j-1} + p_j$.
18:             else
19:                 Assign $J_j$ to $M_1$, and set $L_2^j = L_2^{j-1}$.
20:             end if
21:         end if
22:     end if
23: end if
```

that Algorithm 3 reaches the optimal solution. Otherwise, let $J_l = (p_l, 2)$ be the last job of hierarchy 2 assigned to M_1, and $J_t = (p_t, 2)$ be the first largest job with $p_t = p_{\max,2}$. Clearly, $l \neq t$, as J_t is assigned to M_2.

If $l < t$, by the choice of Algorithm 3, we have $(\sqrt{5}-1)d < L_2^{l-1} + p_{\max,2} + p_l \leq 2(L_2^{l-1} + p_{\max,2})$. Since J_t is assigned to M_2 after job J_l, we have $L_2 \geq L_2^{l-1} + p_{\max,2} > \frac{(\sqrt{5}-1)d}{2}$.

If $l > t$, J_t is assigned to M_2 before assigning job J_l, which implies that $L_2^{l-1} \geq p_t = p_{\max,2} \geq p_l$. Moreover, by the choice of Algorithm 3, we have $L_2^{l-1} + p_l > (\sqrt{5}-1)d$, which implies that $L_2 \geq L_2^{l-1} \geq \frac{(\sqrt{5}-1)d}{2}$. Based on Lemma 1, we have

$$\frac{X^{\mathrm{OPT}}}{X^A} \leq \frac{2d}{d+L_2} \leq \frac{2d}{d + \frac{(\sqrt{5}-1)d}{2}} = \sqrt{5}-1.$$

Case 2: $\max\{L_1, L_2\} = L_2 > d$. Set $X^A = d + L_1$. Let $J_l = (p_l, 2)$ be the last job assigned to M_2. If J_l is assigned to M_2 in Line 16, according to the choice of Algorithm 3, $L_2 = L_2^{l-1} + p_l \leq (\sqrt{5}-1)d$.

If J_l is the first largest job with $p_l = p_{\max,2}$, it is assigned to M_2 in Line 13. Let $J_t = (p_t, 2)$ be the last job assigned to M_2 before J_l. According to the choice of Algorithm 3, we have $L_2 = L_2^{t-1} + p_t + p_l = L_2^{t-1} + p_{\max,2} + p_t \leq (\sqrt{5}-1)d$.

By Lemma 1, we have

$$\frac{X^{\mathrm{OPT}}}{X^A} \leq \frac{T}{d+L_1} = \frac{L_1+L_2}{d+L_1} \leq \frac{(\sqrt{5}-1)d + L_1}{d+L_1} \leq \sqrt{5}-1,$$

which is the desired conclusion. ■

5. Conclusions

In this paper, we studied several online or semi-online scheduling models with the goal of early work maximization under a common due date in a system consisting of two identical hierarchical machines. For each model, we proposed an optimal online or semi-online algorithm, by analysing the problem's lower bound and proving the algorithm's competitive ratio.

Particularly, we studied the pure online model where no information on problem input is known in advance ($P2|GoS, online, d_j = d|\max(X)$), the semi-online models where the maximum processing time is known together with the hierarchy of the largest job ($P2|GoS, online, d_j = d, p_{\max,1}|\max(X)$ and $P2|GoS, online, d_j = d, p_{\max,2}|\max(X)$). For these models, we designed optimal (semi-)online algorithms with competitive ratios of $\sqrt{2}$, $\frac{6}{5}$ and $\sqrt{5}-1$, respectively.

The obtained results demonstrate that the natural directions for future research could be the analysis of some other semi-online variants, such as when the information of the jobs from one of the hierarchies is known; but for the other hierarchy it is completely unknown. Moreover, one could focus on the more complex models, such as extending our models to two (or more) uniform machines (Xiao *et al.*, 2023), scheduling two (or more) types of jobs on an arbitrary number of machines, scheduling jobs with a machine-dependent processing time, or even scheduling jobs with arbitrary due dates.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China (nos. 12371363, 12071417), the Foundation Research Project of the Educational Department of Liaoning Province (LJKZZ20220085), and the statutory funds of the Poznan University of Technology (Poland).

References

Akaria, I. and Epstein, L. (2022). Online scheduling with migration on two hierarchical machines, *Journal of Combinatorial Optimization* **44**(5): 3535–3548.

Bar-Noy, A., Freund, A. and Naor, J. (2001). On-line load balancing in a hierarchical server topology, *SIAM Journal on Computing* **31**(2): 527–549.

Chen, X., Kovalev, S., Liu, Y., Sterna, M., Chalamon, I. and Błażewicz, J. (2021). Semi-online scheduling on two identical machines with a common due date to maximize total early work, *Discrete Applied Mathematics* **290**: 71–78.

Chen, X., Liang, Y., Sterna, M., Wang, W. and Błażewicz, J. (2020a). Fully polynomial time approximation scheme to maximize early work on parallel machines with common due date, *European Journal of Operational Research* **284**(1): 67–74.

Chen, X., Shen, X., Kovalyov, M.Y., Sterna, M. and Błażewicz, J. (2022). Alternative algorithms for identical machines scheduling to maximize total early work with a common due date, *Computers & Industrial Engineering* **171**: 108386.

Chen, X., Sterna, M., Han, X. and Błażewicz, J. (2016). Scheduling on parallel identical machines with late work criterion: Offline and online cases, *Journal of Scheduling* **19**: 729–736.

Chen, X., Wang, W., Xie, P., Zhang, X., Sterna, M. and Błażewicz, J. (2020b). Exact and heuristic algorithms for scheduling on two identical machines with early work maximization, *Computers & Industrial Engineering* **144**: 106449.

Gordon, V., Proth, J.M. and Chu, C. (2002). A survey of the state-of-the-art of common due date assignment and scheduling research, *European Journal of Operational Research* **139**(1): 1–25.

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

Györgyi, P. and Kis, T. (2020). A common approximation framework for early work, late work, and resource leveling problems, *European Journal of Operational Research* **286**(1): 129–137.

Janiak, A., Kwiatkowski, T. and Lichtenstein, M. (2013). Scheduling problems with a common due window assignment: A survey, *International Journal of Applied Mathematics and Computer Science* **23**(1): 231–241, DOI: 10.2478/amcs-2013-0018.

Jiang, Y. (2008). Online scheduling on parallel machines with two GoS levels, *Journal of Combinatorial Optimization* **16**(1): 28–38.

Jiang, Y., Guan, L., Zhang, K., Liu, C., Cheng, T. and Ji, M. (2021). A note on scheduling on two identical machines with early work maximization, *Computers & Industrial Engineering* **153**: 107091.

Jiang, Y., He, Y. and Tang, C. (2006). Optimal online algorithms for scheduling on two identical machines under a grade of service, *Journal of Zhejiang University: Science A* **7**(3): 309–314.

Li, W. (2024). Improved approximation schemes for early work scheduling on identical parallel machines with common due date, *Journal of the Operations Research Society of China* **12**: 341–350, DOI: 10.1007/s40305-022-00402-y.

Liu, X., Wang, W., Chen, X., Sterna, M. and Blazewicz, J. (2023). Exact approaches to late work scheduling on unrelated machines, *International Journal of Applied Mathematics and Computer Science* **33**(2): 285–295, DOI: 10.34768/amcs-2023-0021.

Luo, T. and Xu, Y. (2015). Semi-online hierarchical load balancing problem with bounded processing times, *Theoretical Computer Science* **607**: 75–82.

Park, J., Chang, S. and Lee, K. (2006). Online and semi-online scheduling of two machines under a grade of service provision, *Operations Research Letters* **34**(6): 692–696.

Sterna, M. (2011). A survey of scheduling problems with late work criteria, *Omega* **39**(2): 120–129.

Sterna, M. (2021). Late and early work scheduling: A survey, *Omega* **104**: 102453.

Sterna, M. and Czerniachowska, K. (2017). Polynomial time approximation scheme for two parallel machines scheduling with a common due date to maximize early work, *Journal of Optimization Theory and Applications* **174**: 927–944.

Wang, W., Chen, X., Musial, J. and Blazewicz, J. (2020). Two meta-heuristic algorithms for scheduling on unrelated machines with the late work criterion, *International Journal of Applied Mathematics and Computer Science* **30**(3): 573–584, DOI: 10.34768/amcs-2020-0042.

Xiao, M., Liu, X. and Li, W. (2021). Semi-online early work maximization problem on two hierarchical machines with partial information of processing time, *in* W. Wu and H. Du (Eds), *Algorithmic Applications in Management*, Lecture Notes in Computer Science, Vol. 13153, Springer, Cham, pp. 146–156.

Xiao, M., Liu, X. and Li, W. (2023). Semi-online early work maximization problems on two hierarchical uniform machines with partial information of processing time, *Journal of Combinatorial Optimization* **46**(3): 21.

Man Xiao is currently a doctoral student in the School of Mathematics and Statistics at Yunnan University, China. His research interests include online scheduling and computational geometry.

Xiaoqiao Liu holds an MS degree from the School of Mathematics and Statistics at Yunnan University, China. Her research interests include combinatorial optimization and scheduling problems.

Weidong Li received his PhD degree in the School of Mathematics and Statistics at Yunnan University, China, in 2010. He is currently a full professor there. His research interests include discrete optimization, algorithmic game theory and cloud computing.

Xin Chen received his BS degree in computer science from the Dalian University of Technology, China, in 2005 and his PhD degree in information science from the Poznan University of Technology, Poland, in 2014. He is currently a full professor at the School of Electronic and Information Engineering, Liaoning University of Technology, China. His research covers combinatorial optimization, especially scheduling problems and algorithm design.

Malgorzata Sterna is a full professor at the Poznan University of Technology. Her research interests include scheduling theory, complexity theory, algorithms design, combinatorial optimization and selected aspects of graph theory. She holds a PhD and a habilitation (DSc) in computer science from the Poznan University of Technology.

Jacek Blazewicz is a full professor at the Poznan University of Technology and the director of the Institute of Computing Science there. His research interests include algorithm design, computational complexity, scheduling, combinatorial optimization, bioinformatics, e-commerce. He holds a PhD and a habilitation (DSc) in computer science from the Poznan University of Technology. His publication record includes over 400 papers in many outstanding journals. He is also the author and a co-author of over ten monographs.

Received: 20 October 2023
Revised: 10 January 2024
Accepted: 2 February 2024