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,*, MAŁGORZATA STERNA c, JACEK BLAZEWICZ c,d

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

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

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

dEuropean 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...
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 \( p \)-approximation algorithm \( A \) is a polynomial time algorithm that always produces a feasible solution (for any instance \( I \) with a minimum \( p \)) satisfying \( \frac{C_{OPT}(I)}{C(A, I)} \leq p \), in which \( C_{OPT}(I) \) (\( C^{OPT} \) 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 \( \frac{C_{OPT}}{C^{A}} \leq r \) 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 \( \delta \) means that \( \delta \) 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 \( \delta \), 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 \((m = 2)\), Stern and Czerniackowska (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 \( \frac{\sqrt{5} + 1}{2} \), and then Jiang et al. (2021) demonstrated that the tight bound of LPT is exactly \( \frac{\sqrt{5} - 1}{2} \). 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 \( \frac{2}{3} \). 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 \( \frac{2}{3} \). 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
algorithm with a competitive ratio of \( \min \{1 + s, \frac{2 + \sqrt{2s}}{1 + \sqrt{2s}}\} \) 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 \( \min \{1 + s, \frac{2 + \sqrt{2s}}{1 + \sqrt{2s}}\} \) and \( \min \{s + 1, \sqrt{2s^2 + 2s + 2} - s\} \), respectively. When both mentioned pieces of information are provided, they proved a lower bound \( \frac{2s + 2}{s + 1} \) when \( s \leq \frac{1}{2} \), and designed an optimal online algorithm with a competitive ratio of \( \frac{2s + 2}{s + 1} \) for \( s > \frac{1}{2} \).

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 \( \frac{2s + 2}{s + 1} \) 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.

### Table 1. Relevant results.

<table>
<thead>
<tr>
<th>Model</th>
<th>Tight bound</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pure online</td>
<td>( \sqrt{2} )</td>
<td>This paper</td>
</tr>
<tr>
<td>( T_1 )</td>
<td>( \sqrt{5} - 1 )</td>
<td>Xiao et al., 2021</td>
</tr>
<tr>
<td>( T_2 )</td>
<td>( \sqrt{5} - 1 )</td>
<td>Xiao et al., 2021</td>
</tr>
<tr>
<td>( T_1 &amp; T_2 )</td>
<td>( \frac{6}{5} )</td>
<td>Xiao et al., 2021</td>
</tr>
<tr>
<td>( T )</td>
<td>( \frac{4}{5} )</td>
<td>Xiao et al., 2023</td>
</tr>
<tr>
<td>( p_{max} )</td>
<td>( \sqrt{2} )</td>
<td>This paper</td>
</tr>
<tr>
<td>( p_{max,1} )</td>
<td>( \frac{2}{5} )</td>
<td>This paper</td>
</tr>
<tr>
<td>( p_{max,2} )</td>
<td>( \frac{\sqrt{5} - 1}{4} )</td>
<td>This paper</td>
</tr>
</tbody>
</table>

### 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 \( J = \{J_1, J_2, \ldots, J_n\} \) 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 \( p_j \leq d \) for \( j = 1, 2, \ldots, n \).

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 \( d \) \( (C_j \leq d) \), 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 \( d \) \( (C_j - p_j \geq d) \), 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 \( P2|GoS, online, d_j = d|\max_{X}(X) \), 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 \( S_1 \cap S_2 = \emptyset \). We define the load of machine \( M_i \) \( (i \in \{1, 2\}) \) by \( L_i \), i.e., \( L_i = \sum_{j \in S_i} p_j \). 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 \( L_i \) be the load of \( M_i \) after job \( J_j \) is assigned to one of the machines \( (i \in \{1, 2\}) \) and \( 1 \leq j \leq n \), 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
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 \( X^{\text{OPT}} \) 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 \( X^{\text{OPT}} \) satisfies

\[
X^{\text{OPT}} \leq \min \{T, 2d\} \leq d + \frac{T}{2}.
\]

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 \( \Delta \in \{p_{\text{max}}, p_{\text{max},1}, p_{\text{max},2}\} \) stands for the maximum processing time \( (p_{\text{max}}) \), along with its hierarchy \( k \in \{1, 2\} \) \( (p_{\text{max},1} \text{ and } p_{\text{max},2}) \) (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 \( X^{\text{OPT}} \geq \sqrt{2} \), 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 \( \min \{L_1, L_2\} \geq d \), we have \( X^A = 2d \geq X^{\text{OPT}} \). If \( \max \{L_1, L_2\} \leq d \), we have \( X^A = T \geq X^{\text{OPT}} \). This implies that we only need to consider the case when \( \min \{L_1, L_2\} < d \leq \max \{L_1, L_2\} \), 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 \( X^A = d + L_1 \). 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 \( J_1 = (p_1, 2) \) be the last job of hierarchy 2 assigned to \( M_1 \). By the choice of Algorithm 1, we have \( L_2 \geq L_2^{(i-1)} > (\sqrt{2} - 1)d \). By Lemma 1, we have

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

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 \( J_1 = (p_1, 2) \) be the last job assigned to \( M_2 \).

If \( J_1 \) is assigned to \( M_2 \) in Line 7 of the algorithm, we have \( L_2 \geq L_2^{(i)} = L_2^{(i-1)} + p_i > d \) and \( L_2^{(i-1)} \leq (\sqrt{2} - 1)d \). Therefore, \( T_2 = L_2^{(i-1)} + p_i + L_{1,2} \leq (\sqrt{2} - 1)d + p_i + L_{1,2} \leq
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 $P^2|\text{GoS, online}, d_j = d, \Delta| \max(X)$, 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 $P^2|\text{GoS, online}, d_j = d, p_{\max}| \max(X)$, 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 $P^2|\text{GoS, online}, d_j = d, p_{\max,1}| \max(X)$, we prove a tight bound of $\sqrt{5} - 1$ (cf. Section 4.1). In contrast, if the hierarchy of the largest job is 2 ($\Delta = p_{\max,2}$), i.e., for the problem $P^2|\text{GoS, online}, d_j = d, p_{\max,2}| \max(X)$, 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 $p_j \leq p_{\max,1} \leq d$.

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 $P^2|\text{GoS, online}, d_j = d, p_{\max,1}| \max(X)$ has a competitive ratio of at least $\frac{6}{5}$.

Proof. Let $d = 1$ and $p_{\max,1} = \frac{4}{5}$. The first two jobs are $J_1 = (\frac{4}{5}, 1)$ and $J_2 = (\frac{4}{5}, 2)$. Job $J_1$ can only be assigned to $M_1$.

Case 1: If $J_2$ is assigned to $M_1$, the last job $J_3 = (\frac{4}{5}, 1)$ arrives, implying that $X^A = 1$. However, we have $X^{opt} = \frac{5}{4}$ 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 $J_3 = (\frac{4}{5}, 2)$ arrives. Subsequently, we distinguish the following two subcases:

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

Case 2.2: If $J_3$ is assigned to $M_2$, the next job $J_4 = (\frac{4}{5}, 2)$ arrives.

If $J_4$ is assigned to $M_1$, the last job $J_5 = (\frac{4}{5}, 1)$ arrives, implying that $X^A = \frac{4}{5}$. On the other hand, by the assignment of $\{J_1, J_3\}$ to $M_1$ and $\{J_2, J_3, J_4\}$ to $M_2$, we get $X^{opt} = 2$.

If $J_4$ is assigned to $M_2$, then no more jobs arrive, implying that $X^A = \frac{5}{4}$. In this case, an optimal solution can be get by assigning $\{J_1, J_3\}$ to $M_1$ and $\{J_2, J_3, J_4\}$ to $M_2$, with $X^{opt} = 2$.

Thus, $\frac{X^{opt}}{X^A} \geq 2$ in any case.

Now we propose an optimal semi-online algorithm, i.e., Algorithm 2, to solve the problem $P^2|\text{GoS, online}, d_j = d, p_{\max,1}| \max(X)$. 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 $\frac{2d}{\sqrt{5}}$. 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 $\min\{L_1, L_2\} \geq d$ or $\max\{L_1, L_2\} \leq d$, 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 $J_1 = (p_1, 2)$ be the last job of hierarchy 2 assigned to $M_1$. By the choice of Algorithm 2, we have $L_2 \geq L_2^{-1} \geq \frac{2d}{3}$. Based on Lemma 1, we have

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

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,

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

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

$$\frac{X^{OPT}}{X^A} \leq \frac{T}{d + L_1} = \frac{L_1 + L_2}{d + L_1} = 1 + \frac{L_2 - d}{L_1 + d} \leq 1 + \frac{\alpha \cdot \frac{2d}{3} - d}{L_1 + d} = 1 + \frac{3 \alpha - 1}{3 + 3 \alpha} = 2 - \frac{4}{3 + 3 \alpha} < \frac{6}{5},$$

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 $P2|\text{GoS, online}, d_j = d, p_{\max,2}|$ 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 $J_2 = (\frac{\sqrt{5} - 1}{2}, 1)$ and $J_3 = (\frac{\sqrt{5} - 1}{2}, 1)$ arrive, which implies that $X^A = 1$.

But we can get $X^{OPT} = \frac{\sqrt{5} + 1}{2}$ 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 $J_2 = (\frac{\sqrt{5} - 1}{2}, 1)$ arrives.

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

Thus, $\frac{X^{OPT}}{X^A} \geq \sqrt{5} - 1$ 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 $n_1 = 0$ 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
by Lemma 1, we have
\[
\frac{X^{\text{OPT}}}{X^A} \leq \frac{T}{d + L_1} = \frac{L_1 + L_2}{d + L_1} \leq \frac{(\sqrt{5} - 1)d + L_1}{d + L_1} \leq \sqrt{5} - 1,
\]
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 analyzing 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, \(d_j = d\max(X)\)) and P2|GoS, online, \(d_j = d\max(\max(X))\). For these models, we designed optimal (semi-)online algorithms with competitive ratios of \(\sqrt{5}\), \(\frac{\sqrt{5}}{2}\), 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


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.
Online and semi-online scheduling on two hierarchical machines . . .

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