Improved bounds for online scheduling with eligibility constraints

Kyungkuk Lim\textsuperscript{a}, Kangbok Lee\textsuperscript{b,∗}, Soo Y. Chang\textsuperscript{a}

\textsuperscript{a} Department of Industrial and Management Engineering, Pohang University of Science and Technology, San 31 Hyoja-Dong, Pohang, Kyungbuk, 790-784, Republic of Korea
\textsuperscript{b} Department of Supply Chain Management & Marketing Sciences, Rutgers Business School, 1 Washington Park, Newark, NJ 07102, USA

A R T I C L E   I N F O

Article history:
Received 1 July 2010
Received in revised form 12 May 2011
Accepted 16 May 2011
Communicated by D.-Z. Du

Keywords:
Online scheduling
Parallel machines
Eligibility constraints
Greedy algorithm
Competitive ratio

A B S T R A C T

We consider the online parallel machine scheduling problem of minimizing the makespan under eligibility constraints that restrict each job to be processed only on one of its eligible machines. The greedy approach known as AW is known to be optimal for this problem when the number of machines, \( m \), is a power of 2, i.e. \( m = 2^k \). However, in other cases, the gap between the best known competitive ratio and its lower bound can be as large as 1. In this paper, we construct new competitive ratio and its lower bound whose gap is no more than an irrational number which is approximately 0.1967 and establish optimality for the cases when the number of machines can be written as a sum of two powers of 2, i.e. \( m = 2^k + 2^{k'} \) for \( k \neq k' \). We further analyze the case with seven machines showing that their gap is no more than 1/180 (≈ 0.00556). Moreover, we present new lower bounds of the competitive ratio for the cases with interval and nested eligibility as well as improved competitive ratios for several cases with GoS eligibility.

1. Introduction

We consider the online scheduling of independent jobs on parallel machines where jobs are presented one at a time. Only after a job is presented, its processing time and the set of machines that are eligible to process it are revealed to the scheduler. Then, the scheduler has to immediately and irrevocably assign the job to one of its eligible machines without knowing whether there will be more jobs and, if so, what will be their characteristics. The objective is to minimize the latest completion time of the jobs known as the makespan.

In real world scheduling problems, we often find cases where each job must be processed by one of the eligible machines that are specified for it in advance. Such scheduling problems are referred to as scheduling problems under eligibility constraints. The scheduling under a Grade-of-Service provision [16] and the load balancing with assignment restriction [3–5] are typical examples of such cases. A literature survey on scheduling under eligibility constraints can be found in [25] and another on online scheduling context in [23].

An online algorithm \( \mathcal{A} \) is said to be \( ρ \)-competitive, if for every problem instance \( \mathcal{A} \) is guaranteed to yield a schedule with its makespan never more than \( ρ \) times the optimum makespan of the offline version of the same problem. Such \( ρ \) is referred to as the competitive ratio of the algorithm. On the other hand, we say that an online scheduling problem has lower bound \( ρ' \) of the competitive ratio if no deterministic online algorithm can have a competitive ratio smaller than \( ρ' \). If \( ρ \) and \( ρ' \) so defined are same, the online algorithm is said to be optimal for the particular online problem. Finding an optimal algorithm for a particular online scheduling problem is often an ultimate theoretical inquiry relative to each particular online scheduling problem [2]. Thus, many researchers have provided upper bounds by developing algorithms with the worst case analysis and lower bounds by generating adversary examples while trying to reduce the gap between the two bounds.

∗ Corresponding author. Tel.: +1 917 804 0213.

E-mail addresses: kangbok.lee3@gmail.com, kangblee@business.rutgers.edu (K. Lee).

0304-3975/ – see front matter. Published by Elsevier B.V.
doi:10.1016/j.tcs.2011.05.029
For the problems without eligibility constraints, there have been a number of upper and lower bound results for a competitive ratio on the problem with an arbitrary number of machines or with a specific number of machines. Graham [14] proves that algorithm LS (List Scheduling) has a competitive ratio of $2 - 1/m$ for $m$ machines. Faigle et al. [10] show that LS is optimal for two and three machines and $1 + 1/\sqrt{2} (\approx 1.7071)$ is the lower bound to the competitive ratio of any algorithm for four or more machines. Chen et al. [8] develop a slightly improved lower bound for ten or less machines. Baral et al. [6] prove that no deterministic online algorithm can have a competitive ratio smaller than 1.837, for $m \geq 3454$. For any number of machines, Albers [1] prove that 1.852 is a lower bound to the competitive ratio of any algorithm. The best lower bound currently known is due to Rudin [33]. He prove that no deterministic online algorithm can be better 1.88-competitive. Rudin and Chandrasekaran [34] prove that no deterministic online algorithm can have a competitive ratio smaller than $\sqrt{3} (\approx 1.7321)$ for four machines. For the upper bound, Galambos and Woeginger [12] present an algorithm that is $(2 - 1/m - \varepsilon_m)$-competitive, where $\varepsilon_m > 0$, but $\varepsilon_m$ tends to 0 as $m$ goes to infinity. Bartal et al. [7] design an algorithm whose performance guarantee is asymptotically smaller than 2. Their algorithm is $2 - 1/70 (\approx 1.986)$-competitive, for all $m \geq 70$. Later, Karger et al. [20] prove a competitive ratio of 1.945 for any number of machines. Albers [1] design an algorithm whose competitive ratio is 1.923. Finally, Fleischer and Wahl [11] design an algorithm with a competitive ratio smaller than 1.9201 when the number of machines tends to infinity.

For the problem subject to eligibility constraints, Azar et al. [4] present a greedy type algorithm known as $AW$ which assigns each job to the least loaded machine among eligible machines while breaking ties arbitrarily. The competitive ratio of algorithm $AW$ is proved to be $\lceil \log_2 m \rceil + 1$ for the case with $m$ machines. They also prove that the lower bound of the problem is $\lceil \log_2 (m + 1) \rceil$. Since this lower bound meets the competitive ratio whenever $m$ is a power of 2, algorithm $AW$ is optimal in cases with $m = 2^k$. However, when $m$ is not a power of 2, particularly when $m = 2^k - 1$, the gap between the competitive ratio and its lower bound may get as large as 1. Later, Hwang et al. [15] develop a slightly improved competitive ratio of algorithm $AW$ which is $\log_2 m + 1$. Nevertheless, the gap still approaches 1 for the case with $m = 2^k - 1$ as $k$ increases.

Unlike general eligibility having arbitrary non-empty subsets of machine sets as eligible sets, some eligibilities have a structured property [23]. One example is Grade-of-Service (GoS) eligibility which is introduced by Hwang et al. [16]. Let $E_j$ denote the eligible set of job $j$. GoS eligibility implies that for any pair of jobs $j$ and $k$, $E_j \subseteq E_k$ or $E_j \supseteq E_k$. This eligibility can be observed in a great deal of literature in the context of offline scheduling [13,18,26,30] and online scheduling [5,19,21,31]. Another structured eligibility is nested eligibility which is introduced by Pinedo [32]. It means that, for two jobs $j$ and $k$, $E_j \subseteq E_k$ or $E_j \supseteq E_k$ or $E_j \cap E_k = \emptyset$. This eligibility has been discussed in the context of offline scheduling [13,17,29] and online scheduling [22]. We also consider another called the interval eligibility which is more general than GoS eligibility and nested eligibility. The interval eligibility means that the eligible set of a job can be expressed by a set of consecutive machines in a linear ordering of machines. Formally, let $\mu_j$ and $\nu_j$ denote the first and last eligible machine indices, respectively, i.e., $E_j = \{ M_i \in M | \mu_j \leq i \leq \nu_j \}$ where $M$ is the set of all machines and $M_i$ is the $i$th machine. There are a few research results on this eligibility; for offline problem Choi et al. [9] and Lee et al. [24] deal with this eligibility and Lin and Li [28] call this eligibility convex eligibility and for online problem Bar-Noy et al. [5] deal with this eligibility. Among these eligibilities, we can see the following relationship [23].

GoS $\subset$ nested $\subset$ interval $\subset$ general.

Note that when the number of machines is two, general eligibility, interval eligibility and nested eligibility are identical to each other. However, the number of machines is more than two, they are different from each other.

The online scheduling under a class of the eligibility which certainly encompasses that the cases of GoS eligibility is analyzed by Bar-Noy et al. [5]. The authors only consider the case of the infinite number of machines and present an $(e + 1)$-competitive algorithm and a lower bound example implying that any online algorithm cannot give a competitive ratio less than $e$ where $e$ is the base of the natural logarithm. However, the question that still remains to be answered is whether there are better online algorithms for a finite number of machines under GoS eligibility. For the online scheduling on two machines with GoS eligibility, Park et al. [31] develop an algorithm that is shown to be optimal with a competitive ratio of 5/3. For three machine case, Lim et al. [27] present an online algorithm which is also optimal with a competitive ratio of 2. Recently, Tan and Zhang [35] improve the competitive ratio by using a fractional solution that is obtained from a linear programming formulation. Moreover, they provide another online algorithm for the cases of four and five machines with better competitive ratios of 2.333 and 2.610, respectively. Still, there is a gap between the lower bound and the upper bound of competitive ratio for $m \geq 4$. For nested eligibility, although there are several results in the context of offline scheduling, there are no research results on online scheduling. Online scheduling under interval eligibility is discussed by Bar-Noy et al. [5]. The authors provide a lower bound example with a competitive ratio of $1/\log m$ where $m$ is a power of two. Thus, the competitive ratio for the cases of nested eligibility and interval eligibility has not been fully considered.

In this paper, we construct a new competitive ratio for algorithm $AW$ and a lower bound of the competitive ratio for the problem with general eligibility constraints and we show that their gap is no more than an irrational number which is approximately 0.1967. Also, we establish optimality for the cases when the number of machines can be written as a sum of two powers of 2, i.e. $m = 2^k + 2^{k'}$ for some $k \neq k'$. This result implies that seven machine case is the first case that is not yet proved whether $AW$ is indeed optimal or not. Hence, we further analyze the case with seven machines showing that the gap is no more than $1/180 (\approx 0.00556)$ by improving both the competitive ratio for algorithm $AW$ and the lower
bound. Moreover, we present several lower bound results of the competitive ratio for the problems with different number of machines and interval and nested eligibility constraints. In addition, we provide an algorithm for online scheduling problem with GoS eligibility constraints when the number of machines is 4 and 5.

We define notations for precise description of the problem. Unknown number of jobs denoted by \( J = \{1, 2, \ldots, n\} \) are to be scheduled on \( m \) parallel machines denoted by \( M = \{M_1, M_2, \ldots, M_m\} \). The \( j \)th job is referred to as job \( j \). For each job \( j \), the time required to process it, denoted by \( p_j \), and the subset of machines that are eligible to process it, denoted by \( E_j \), are not known until the job is presented. We define \( l(j) \) to be the index of machine on which job \( j \) is assigned in the lower bound example. \( t(j) \) denotes \( \sum_{j \in J'} p_j \) if \( J' \subset J \) is non-empty, and 0 otherwise. \( z^\ast \) and \( z^\ast' \) denote the makespan of the schedule generated by algorithm \( A \) and the optimum makespan, respectively. Then the competitive ratio of \( A \) is defined to be the maximum of \( z^\ast/z^\ast' \).

The remaining part of the paper is organized as follows. In Section 2, we develop improved competitive ratio and its lower bound for general eligibility case and further improve the case with seven machines. In Section 3, we discuss lower bounds of the cases with nested and interval eligibility constraints and the competitive ratio of the case with GoS eligibility constraints. Conclusion and future research directions are provided in Section 4.

2. General eligibility constraints

In this section, we present improved competitive ratio and its lower bound whose gap is no more than an irrational number which is approximately 0.1967. It is known that algorithm \( A \) is optimal for the case where the number of machines can be written as a power of 2, i.e., \( m = 2^k \) for a non-negative integer \( k \). The new lower and upper bounds reveal that \( A \) is also optimal for the case where the number of machines can be written as a sum of two powers of 2, i.e., \( m = 2^k + 2^{k'} \) for non-negative integers \( k \) and \( k', k \neq k' \).

For the analysis of \( A \), we let \( S_i \) be the set of jobs that are scheduled on machine \( M_i \), and \( S'_j \) be the set of jobs assigned to machine \( M_j \) right after job \( j \) is scheduled by \( A \). Also, we let \( S^*_i \) be the set of jobs assigned to machine \( M_i \) in the optimal schedule, and \( S^*_k \) be a set of jobs in \( S_i \) which appear in \( S^*_k \) (i.e., \( S^*_k = S_i \cap S^*_k \)). Finally, we define

\[
R_k = \sum_{i=1}^{m} \max\{0, t(S_i) - kz^\ast\}
\]

for an arbitrary non-negative integer \( k \). Finally, we note

\[
R_0 = \sum_{j \in J} p_j \quad \text{and} \quad \frac{R_0}{m} = \frac{\sum_{j \in J} p_j}{m} \leq z^\ast.
\]

2.1. Improved lower bound

**Theorem 1.** The lower bound of our problem with \( m \) machines, denoted by \( LB(m) \), can be written as the following recursive formula

\[
LB(1) = 1 \quad \text{and} \quad LB(m) = LB(y_m) + \frac{1}{\left[ \frac{y_m}{m - y_m} \right]} \quad \text{for} \ m \geq 2
\]

where \( y_m = \arg\max_{\{2 \leq i \leq m-1\}} \left\{ LB(i) + \frac{1}{\left[ \frac{i}{m-i} \right]} \right\} \).

**Proof.** The procedure \( LBEX(m) \) presented in Fig. 1 generates a lower bound example, having the optimal makespan of 1, for the case with \( m \) machines. This procedure interacts with an imaginary algorithm that assigns each job \( j \) to \( M_{l(j)} \in E_j \). Seeing how \( y_m \) is determined is quite crucial to understanding how the procedure \( LBEX(m) \) works.

With familiarity with the definition of \( y_m \) and \( LB(m) \), it can easily be seen that \( LBEX(m) \) uses a lower bound example for the case with \( y_m \) machines. In the first while-loop, \( LBEX(m) \) produces \( y_m \) jobs with processing time of \( \left[ \frac{y_m}{m - y_m} \right]^{-1} \). From the second while-loop to the last while-loop, all jobs are eligible at most to the machines where the first \( y_m \) jobs are assigned and all jobs except the first \( y_m \) jobs can be regarded as the lower bound example for the case with \( y_m \) machines. Thus, \( LBEX(m) \) produces a series of jobs that every possible deterministic algorithm yields a schedule whose makespan is equal to \( LB(m) \). Fig. 2(a) illustrates the case with 10 machines.

What is left to be verified is that the optimum makespan of the schedule for the series of jobs generated by \( LBEX(m) \) is 1. To this end, we focus our attention to the execution of the while-loop in \( LBEX(m) \). Fig. 2(b) illustrates the optimum schedule for the case with 10 machines.

In general, each round of the while-loop generates a set of \( y_m \) jobs with an identical length of \( \left[ \frac{y_m}{m - y_m} \right]^{-1} \) which is never more than \( \frac{\tilde{m}}{y_m} \). Hence, within the makespan of 1, these \( y_m \) jobs can be scheduled on \( \tilde{m} - y_m \) machines to which no jobs are assigned during the particular round of the while-loop. Hence, we see that the theorem follows. \( \square \)
LBEX(m)

\[ j = 0; \tilde{M} = \{M_1, M_2, \ldots, M_m\}; \tilde{m} = |\tilde{M}|; \]

**Do While** (\( \tilde{m} > 1 \))

\[ \{ \begin{align*}
g & = \gamma; \\
p & = \lceil \gamma / (\tilde{m} - \gamma) \rceil - 1; \\
\tilde{M} & = \tilde{M} \setminus \{M_i(i)\}; \\
\tilde{M} & = \tilde{M} \cup \{M_i(i)\}; \\
k & += \; & \} \]

\[ \} \]

**j++**: Releases the last job \( j \) with \( p_j = 1 \) and \( E_j = \tilde{M} \);

---

**Fig. 1.** The lower bound example generation.

<table>
<thead>
<tr>
<th>M_1</th>
<th>1</th>
<th>9</th>
<th>13</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_2</td>
<td>2</td>
<td>10</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_3</td>
<td>3</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_4</td>
<td>4</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_7</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_8</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_{10}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) The schedule by algorithm \( A W \).

<table>
<thead>
<tr>
<th>M_1</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_2</td>
<td>15</td>
</tr>
<tr>
<td>M_3</td>
<td>14</td>
</tr>
<tr>
<td>M_4</td>
<td>13</td>
</tr>
<tr>
<td>M_5</td>
<td>12</td>
</tr>
<tr>
<td>M_6</td>
<td>11</td>
</tr>
<tr>
<td>M_7</td>
<td>10</td>
</tr>
<tr>
<td>M_8</td>
<td>9</td>
</tr>
<tr>
<td>M_9</td>
<td>5</td>
</tr>
<tr>
<td>M_{10}</td>
<td>1</td>
</tr>
</tbody>
</table>

(b) The optimal schedule.

---

**Fig. 2.** The lower bound example for ten machines.

2.2. The competitive ratio of algorithm \( A W \)

Now, we develop a new competitive ratio of algorithm \( A W \). Before analyzing the algorithm in detail, we note the following theorem and its corollary established by Azar et al. [4] without proofs.

**Theorem 2.** \( R_{u-1} \geq 2R_u \), where, \( u \) is a positive integer [4].

**Corollary 3.** For a positive integer \( u \),

\[ \frac{m}{2^u}z^* \geq R_u. \]

Now we extend the idea by Azar et al. [4] in the following theorem.

**Theorem 4.** Algorithm \( A W \) achieves a competitive ratio of

\[ [\log_2 m] + \frac{m}{2^{[\log_2 m]}}. \]

**Proof.** For \( k = [\log_2 m] \), by Azar et al. [4], \( z^{AW} \leq kz^* + R_k \). By Theorem 2 and the definition of \( R_0 \), we obtain that \( R_k \leq \frac{1}{2^k}R_0 = \frac{1}{2^k} \sum_{j \in J} p_j \). Therefore, since \( k = [\log_2 m] \) and \( \sum_{j \in J} p_j \leq mz^* \),

\[ z^{AW} \leq kz^* + R_k \leq kz^* + \frac{1}{2^k} \sum_{j \in J} p_j \leq \left( [\log_2 m] + \frac{m}{2^{[\log_2 m]}} \right) z^*. \]

Recall that Azar et al. [4] prove that if the number of machines is \( 2^k \) for some positive integer \( k \), then algorithm \( A W \) is optimal with the competitive ratio of \( k + 1 \). We extend this idea to other general cases.

**Theorem 5.** If the number of machines is expressed by \( 2^k + 2^{k'} \), where \( k' < k \) are non-negative integers, then algorithm \( A W \) is optimal.
For the cases with seven machines, the competitive ratio of any deterministic algorithm can never be less than $3 + 25/36 \approx 3.6944$.

Proof. We construct a problem instance such that the optimal makespan is 36 while any algorithm has the makespan of at least $3 \times 36 + 25$ with following information of the jobs (see Table 2 and Fig. 3).

It needs to be verified that all the jobs can be rearranged so that the makespan is 36. The first 6 jobs are assigned to a machine that belongs to a set $E_6 \setminus \{M_t(6)\}$. The next 3 jobs are assigned to a machine that belongs to a set $E_9 \setminus \{M_t(9)\}$. The next 2 jobs are assigned to a machine that belongs to a set $E_{10} \setminus \{M_t(10)\}$. The job 12 and job 13 are assigned to a machine that belongs to a set $E_{12} \setminus \{M_t(12)\}$ and $E_{13} \setminus \{M_t(13)\}$, respectively. The job 14 is assigned to a machine that belongs to a set $E_{14} \setminus \{M_t(14)\}$. The last job is assigned to machine $M_t(14)$. \( \square \)

Now, we consider the competitive ratio of the algorithm $AW$ for the cases with seven machines is 3.7.

Theorem 8. For the cases with seven machines, $z^{AW} \leq 3.7z^*$. 

Proof. Refer to the online companion to this paper. \( \square \)

By Theorems 7 and 8, the gap between the best known competitive ratio and its lower bound for the case with seven machines is $1/180 \approx 0.00556$. 

\begin{table}
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\rho'$</td>
<td>$\rho$</td>
<td>$\rho$</td>
</tr>
<tr>
<td>2</td>
<td>$2^1$</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>$2^1 + 2^2$</td>
<td>2</td>
<td>3</td>
<td>$\log_2 3 + 1 \approx 2.5850$</td>
</tr>
<tr>
<td>4</td>
<td>$2^2$</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>$2^2 + 2^3$</td>
<td>3</td>
<td>4</td>
<td>$\log_2 5 + 1 \approx 3.3219$</td>
</tr>
<tr>
<td>6</td>
<td>$2^2 + 2^3$</td>
<td>3</td>
<td>4</td>
<td>$\log_2 6 + 1 \approx 3.5850$</td>
</tr>
<tr>
<td>7</td>
<td>$2^2 + 2^3 + 2^4$</td>
<td>3</td>
<td>4</td>
<td>$\log_2 7 + 1 \approx 3.8074$</td>
</tr>
<tr>
<td>8</td>
<td>$2^3$</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>$2^3 + 2^4$</td>
<td>4</td>
<td>5</td>
<td>$\log_2 9 + 1 \approx 4.1699$</td>
</tr>
<tr>
<td>10</td>
<td>$2^3 + 2^4$</td>
<td>4</td>
<td>5</td>
<td>$\log_2 10 + 1 \approx 4.3219$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\rho'$</td>
<td>$\rho$</td>
<td>$\rho$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\rho'$</td>
<td>$\rho$</td>
<td>$\rho$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\rho'$</td>
<td>$\rho$</td>
<td>$\rho$</td>
</tr>
</tbody>
</table>
\end{table}
For any deterministic algorithm in the online scheduling of three machines under interval eligibility, we can consider the online scheduling problems on a small number of machines under interval and nested eligibility constraints. The competitive ratio of algorithm \( \mathcal{A} \) for such problems can be considered \( 1 + \lfloor \log_2 m \rfloor \) since Hwang et al. [16] provide an example whose competitive ratio of \( 1 + \lfloor \log_2 m \rfloor \) for the problem with GoS eligibility and GoS eligibility is a special case of nest and interval eligibility. Thus, for the problems with nested and interval eligibility, we establish the lower bounds to the competitive ratio for the case with different number of machines. Moreover, we present an improved algorithm for the case with GoS eligibility.

### 3.1. Interval eligibility constraints

**Lemma 9.** For any deterministic algorithm \( \mathcal{A} \) for online scheduling of three machines under interval eligibility,

\[
\frac{z^A}{z^*} \geq 1 + \sqrt{2} \approx 2.414.
\]

**Proof.** Generate two jobs with \( p_1 = p_2 = 1 \) and \( E_1 = E_2 = \{M_1, M_2, M_3\} \).

**Case 1.** If two jobs are assigned to the same machine, we generate job 3 with \( p_3 = 1 \) and \( E_3 = \{M_{(2)}\} \), thus, \( \frac{z^A}{z^*} = 3 > 1 + \sqrt{2} \).

**Case 2.** If two jobs are assigned to consecutive machines, we generate job 3 with \( p_3 = 2 \) and \( E_3 = \{M_{(1)}, M_{(2)}\} \). Then, we generate job 4 with \( p_4 = 2 \) and \( E_4 = \{M_{(3)}\} \), thus, \( \frac{z^A}{z^*} = 5/2 > 1 + \sqrt{2} \).

**Case 3.** If two jobs are assigned to non-consecutive machines, implying one job is assigned to machine \( M_1 \) and the other is assigned to machine \( M_3 \), then we generate job 3 with \( p_3 = 1 + \sqrt{2} \) and \( E_3 = \{M_1, M_2, M_3\} \).

**Case 3.1.** If job 3 is assigned to a machine in \( \{M_1, M_3\} \), we generate job 4 with \( p_4 = 1 + \sqrt{2} \) and \( E_4 = \{M_{(3)}\} \). Then \( z^* = 1 + \sqrt{2} \) and \( z^A = 1 + 2(1 + \sqrt{2}) \), thus, \( \frac{z^A}{z^*} = 2 + \frac{1}{1 + \sqrt{2}} = 1 + \sqrt{2} \).

**Case 3.2.** If job 3 is assigned to machine \( M_2 \), we generate job 4 with \( p_4 = \sqrt{2} \) and \( E_4 = \{M_1, M_2, M_3\} \).

**Case 3.2.1.** If job 4 is assigned to a machine in \( \{M_1, M_3\} \), we generate job 5 with \( p_5 = 3 + 2\sqrt{2} \) and \( E_5 = \{M_{(4)}, M_2\} \). Then, we generate job 6 with \( p_6 = 3 + 2\sqrt{2} \) and \( E_6 = \{M_{(5)}\} \). Clearly, \( z^* = 3 + 2\sqrt{2} \) and \( z^A = 1 + \sqrt{2} + 2(3 + 2\sqrt{2}) \), thus, \( \frac{z^A}{z^*} = 2 + \frac{1 + \sqrt{2}}{3 + 2\sqrt{2}} = 1 + \sqrt{2} \).
Case 3.2.2. If job 4 is assigned to machine $M_2$, we generate job 5 with $p_5 = 2\sqrt{2}$ and $E_5 = \{M_1, M_2, M_3\}$.

Case 3.2.2.1. If job 5 is assigned to a machine in $\{M_1, M_3\}$, we generate job 6 with $p_6 = 3 + 4\sqrt{2}$ and $E_6 = \{M_1(5), M_2\}$. Then we generate job 7 with $p_7 = 3 + 4\sqrt{2}$ and $E_7 = \{M_1(6)\}$. Clearly, $z^* = 3 + 4\sqrt{2}$ and $z^A = 1 + 2\sqrt{2} + 2(3 + 4\sqrt{2})$, thus, $\frac{z^A}{z^*} = 2 + \frac{1 + 4\sqrt{2}}{3 + 4\sqrt{2}} > 1 + \sqrt{2}$.

Case 3.2.2.2. If job 5 is assigned to machine $M_2$, we generate job 6 with $p_6 = 4\sqrt{2}$ and $E_6 = \{M_1, M_2, M_3\}$.

Case 3.2.2.2.1. If job 6 is assigned to a machine that belongs to a set $\{M_1, M_3\}$, we generate job 7 with $p_7 = 3 + 8\sqrt{2}$ and $E_7 = \{M_1(6), M_2\}$. Then, we generate job 8 with $p_8 = 3 + 8\sqrt{2}$ and $E_8 = \{M_1(7)\}$. Clearly, $z^* = 3 + 8\sqrt{2}$ and $z^A = 1 + 4\sqrt{2} + 2(3 + 8\sqrt{2})$, thus, $\frac{z^A}{z^*} = 2 + \frac{1 + 4\sqrt{2}}{3 + 8\sqrt{2}} > 1 + \sqrt{2}$.

Case 3.2.2.2.2. If job 6 is assigned to machine $M_2$, we generate job 7 with $p_7 = 3 + 3\sqrt{2}$ and $E_7 = \{M_2\}$. Clearly, $z^* = 3 + 3\sqrt{2}$ and $z^A = 1 + 8\sqrt{2} + (3 + 3\sqrt{2})$, thus, $\frac{z^A}{z^*} = 1 + \frac{1 + 8\sqrt{2}}{3 + 3\sqrt{2}} > 1 + \sqrt{2}$.

In the online companion to this paper, we provide the proofs of Lemmas 10 and 11.

**Lemma 10.** For any deterministic algorithm $\mathcal{A}$ for online scheduling of four machines under interval eligibility,

$$\frac{z^A}{z^*} \geq \frac{3 + \sqrt{3}}{2} \approx 2.618.$$  

**Proof.** Refer to the online companion to this paper. □

**Lemma 11.** For any deterministic algorithm $\mathcal{A}$ for online scheduling of five machines under interval eligibility,

$$\frac{z^A}{z^*} \geq 3.$$  

**Proof.** Refer to the online companion to this paper. □

3.2. Nested eligibility constraints

**Lemma 12.** For any deterministic algorithm $\mathcal{A}$ for online scheduling of three machines under nested eligibility,

$$\frac{z^A}{z^*} \geq 2 + \frac{1}{3}.$$  

**Proof.** Suppose that possible eligible sets are following: $\{M_1\}, \{M_2\}, \{M_3\}, \{M_1, M_2\}$ and $\{M_1, M_2, M_3\}$.

**Case 1.** If job 1 is assigned to machine $M_1$, we generate job 2 with $p_2 = 1/2$ and $E_2 = \{M_1, M_2, M_3\}$.

**Case 1.1.** If job 2 is assigned to machine $M_1$, we generate job 3 with $p_3 = 1$ and $E_3 = \{M_1\}$, thus, $\frac{z^A}{z^*} = 2 + 1/2$.

**Case 1.2.** If job 2 is assigned to machine $M_2$, we generate job 3 with $p_3 = 3/2$ and $E_3 = \{M_1, M_2\}$. Then, we generate job 4 with $p_4 = 3/2$ and $E_4 = \{M_1(3)\}$, thus, $\frac{z^A}{z^*} \geq 2 + 1/3$.

**Case 1.3.** If job 2 is assigned to machine $M_3$, we generate job 3 with $p_3 = 3/2$ and $E_3 = \{M_1, M_2, M_3\}$.

**Case 1.3.1.** If job 3 is assigned to a machine in $\{M_1, M_3\}$, we generate job 4 with $p_4 = 3/2$ and $E_4 = \{M_1(3)\}$, thus, $\frac{z^A}{z^*} \geq 2 + 1/3$.

**Case 1.3.2.** If job 3 is assigned to machine $M_2$, we generate job 4 with $p_4 = 3$ and $E_4 = \{M_1, M_2\}$. Then, we generate job 5 with $p_5 = 3$ and $E_5 = \{M_1(4)\}$, thus, $\frac{z^A}{z^*} \geq 2 + 1/3$.

**Case 2.** If job 1 is assigned to machine $M_2$, this case is symmetric to **Case 1.**

**Case 3.** If job 1 is assigned to machine $M_3$, we generate job 2 with $p_2 = 2$ and $E_2 = \{M_1, M_2, M_3\}$.

**Case 3.1.** If job 2 is assigned to machine $M_1$, this case is identical to **Case 1.3** by scaling.

**Case 3.2.** If job 2 is assigned to machine $M_2$, this case is symmetric to **Case 3.1**.

**Case 3.3.** If job 2 is assigned to machine $M_3$, we generate job 3 with $p_3 = 2$ and $E_3 = \{M_3\}$, thus, $\frac{z^A}{z^*} = 2 + 1/2$. □

**Lemma 13.** For any deterministic algorithm $\mathcal{A}$ for online scheduling of four machines under nested eligibility,

$$\frac{z^A}{z^*} \geq 1 + \sqrt{2}.$$  

**Proof.** Refer to the online companion to this paper. □

**Lemma 14.** For any deterministic algorithm $\mathcal{A}$ for online scheduling of five machines under nested eligibility,

$$\frac{z^A}{z^*} \geq 2.5.$$  

**Proof.** Refer to the online companion to this paper. □
Lemma 15. For any deterministic algorithm $A$ for online scheduling of six machines under nested eligibility,

$$\frac{Z^A}{Z^*} \geq 3.$$

**Proof.** Refer to the online companion to this paper. □

Lemma 16. For any deterministic algorithm $A$ for online scheduling with $m$ parallel machines under nested eligibility,

$$\frac{Z^A}{Z^*} \geq 1 + \frac{1}{2} \lfloor \log_2 m \rfloor.$$

**Proof.** Refer to the online companion to this paper. □

3.3. GoS eligibility constraints

Tan and Zhang [35] provide a linear programming based algorithm $DF$ and present another algorithm $HT$ that has better competitive ratios for the cases with four and five machines. As a natural extension of the result for the two machine case by Park et al. [31], we consider a parametric algorithm $\text{Largest Grade Fit, LGF}(\rho_m, m)$ for the problem with $m$ machines and a competitive ratio $\rho_m$.

The following table shows the best known lower bound (LB) of the competitive ratio and the competitive ratios of various algorithms according to the number of machines, $m$. It was proved that $\text{LGF}(5/3, 2)$ [31] and $\text{LGF}(2, 3)$ [27] are optimal for the problems with two and three machines, respectively. This paper shows that $\text{LGF}$ outperforms existing algorithms for $m = 4$ and 5.

<table>
<thead>
<tr>
<th>$m$</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB</td>
<td>1.667</td>
<td>2.000</td>
<td>2.000</td>
<td>2.000</td>
</tr>
<tr>
<td>DF [35]</td>
<td>-</td>
<td>2.500</td>
<td>2.630</td>
<td>2.713</td>
</tr>
<tr>
<td>HT [35]</td>
<td>-</td>
<td>2.000</td>
<td>2.333</td>
<td>2.610</td>
</tr>
<tr>
<td>LGF</td>
<td>1.667</td>
<td>2.000</td>
<td>2.294</td>
<td>2.501</td>
</tr>
</tbody>
</table>

$\text{LGF}(\rho_m, m)$

Step 1 Initialization

- $P_{max} = 0.$
- **FOR** $i = 1$ **TO** $m$, $L_i = 0$

Step 2 When job $j$ is presented

- $P_{max} \leftarrow \max\{p[j], P_{max}\}$
- $L_{g_j} \leftarrow L_{g_j} + p_j$
- $L \leftarrow \max\left\{\max_{1 \leq i \leq m} \left\lfloor \frac{\sum_{k=1}^{i} t_k}{i} \right\rfloor, P_{max}\right\}$
- $k \leftarrow g_j$
- **WHILE** $(k > 0)$
  - **IF** $t(S_k) + p_j \leq \rho_m L$
    - **THEN** $S_k \leftarrow S_k \cup \{j\}$ **GO TO** Step 2.
  - **ELSE** $k \leftarrow k - 1$
  - **IF** $k = 0$, $S_1 \leftarrow S_1 \cup \{j\}$ **return** FALSE.

Let $S'_j$ be the set of jobs scheduled on machine $M_i$ after job $j$ is scheduled and $L'$ be the $L$ value after job $j$ is presented. Note that $L'$ is always a lower bound of the optimum makespan when jobs $1, \ldots, j$ are presented. Let $J_i = \{j \mid g_j = i, j \in J\}$ for $i = 1, 2, \ldots, m$. Moreover, $\rho_m$ is defined for $m = 4, 5$:

- $\rho_4$ is a solution of $13\rho^5 - 217\rho^4 + 1446\rho^3 - 4782\rho^2 + 7821\rho - 5049 = 0$.
- $\rho_5$ is a solution of $68\rho^5 - 1360\rho^4 + 10785\rho^3 - 42125\rho^2 + 80745\rho - 60613 = 0$.

Then, $\rho_4 \approx 2.2942$ and $\rho_5 \approx 2.50081$.

By definition of $L'$, we have

$$ml' \geq \sum_{j=1}^{m} t(S'_j). \quad (1)$$

Lemma 17. For the online scheduling problem with $m$ parallel machines subject to GoS eligibility, by $\text{LGF}(\rho_m, m)$, job $j$ with $g_j$ for $g_j \geq 4$ cannot be assigned to machine $M_i$ for $1 \leq i \leq g_j - 3$ for $m = 4$ and 5 where $\rho_4 \approx 2.2942$ and $\rho_5 \approx 2.50081$. 

Proof. Suppose that job $j$ with $g_j$ for $g_j \geq 4$ is assigned to machine $M_i$ for $1 \leq i \leq g_j - 3$.

\[
\sum_{i=g_j-2}^{g_j} t(S^{i-1}_j) + 3p_j > 3\rho_m \times L^j
\]

\[
mL^j \geq \sum_{i=1}^{m} t(S^{i-1}_j) + p_j
\]

\[2L^j \geq 2p_j.
\]

By adding up both sides, we can conclude a clear contradiction. It completes the proof. □

Corollary 18. For the online scheduling problem with $m$ parallel machines subject to GoS eligibility, by $\text{LGF}(\rho_m, m)$ for $m = 4$ and $5$,

- job $j$ with $g_j = 4$ cannot be assigned to machine $M_1$ when $m = 4$ and $5$;
- job $j$ with $g_j = 5$ cannot be assigned to machine $M_2$ when $m = 5$.

Proof. It immediately holds by Lemma 17. □

Lemma 19. Let job $j$ be eligible to machine $M_i$, for $i \geq 2$. If job $j$ is scheduled on machine $M_{i-1}$ by algorithm $\text{LGF}(\rho_m, m)$, then we have

\[
t(S^i_j) = t(S^{i-1}_j) > (\rho_m - 1)L^j.
\]

Proof. Suppose that job $j$ is scheduled on machine $M_{i-1}$ for $i \geq 2$ by algorithm $\text{LGF}(\rho_m, m)$. Then, we have

\[
t(S^{i-1}_j) + p_j > \rho_m L^j.
\]

Since $p_j \leq L^j$, we obtain

\[
t(S^i_j) = t(S^{i-1}_j) > \rho_m L^j - p_j \geq (\rho_m - 1)L^j.
\]

□

Corollary 20. If job $j$ eligible to machine $M_i$ for $i \geq 3$ is scheduled on machine $M_{i-2}$ by algorithm $\text{LGF}(\rho_m, m)$, we obtain that

\[
t(S^{i-1}_j) + t(S^i_j) > 2(\rho_m - 1)L^j.
\]

Proof. By Lemma 19, we have

\[
t(S^{i-1}_j) + t(S^i_j) > (\rho_m - 1)L^j + (\rho_m - 1)L^j = 2(\rho_m - 1)L^j.
\]

□

Lemma 21. If job $j$ with $g_j = 4$ is scheduled on machine $M_2$ by algorithm $\text{LGF}(\rho_m, m)$, we obtain that

\[
(m + 2 - 2\rho_m)L^j > t(S^4_j) + t(S^3_j).
\]

Proof. By Corollary 20, we have

\[
t(S^4_j) + t(S^3_j) > 2(\rho_m - 1)L^j.
\]

Since

\[
mL^j \geq \sum_{i=1}^{m} t(S^i_j) \geq \sum_{i=1}^{4} t(S^i_j)
\]

and (2), we obtain

\[
(m + 2 - 2\rho_m)L^j > t(S^4_j) + t(S^3_j).
\]

□

Theorem 22. For the online scheduling problem with $m$ parallel machines subject to GoS eligibility, $\text{LGF}(\rho_m, m)$ yields a schedule with its makespan that is at most $\rho_m$ times the optimal makespan for $m = 4$ and $5$ where $\rho_4 \approx 2.2942$ and $\rho_5 \approx 2.50081$.

Proof. Suppose that the theorem does not hold. Then, there must exist a counterexample and we consider a counterexample of which the number of jobs is the minimum, called a minimum counterexample with the last job $n$. If the makespan is determined at machine $M_i$ for $i \neq 1$, it is always less than or equal to $\rho_m$ times the optimal makespan. So, we focus on the case where the makespan is determined at machine $M_1$. Then, we have

\[
t(S^n_i) > \rho_m L^n.
\]

We consider two cases concerning the eligible set of job $n$. 

\[5219\]

K. Lim et al. / Theoretical Computer Science 412 (2011) 5211–5224
Case 1: $g_n \geq 2$

By Lemma 19, this implies that
\[ t(S^1_n) = t(S^1_{n-1}) > (\rho_m - 1)L^n. \]

Thus, by (3), we have that
\[ t(S^1_n) + t(S^2_n) > \rho_m L^n + (\rho_m - 1)L^n = (2\rho_m - 1)L^n. \]  
(4)

Since $t(j_1) + t(j_2) + t(j_3) \leq 3L^n < (2\rho_m - 1)L^n$, by Corollary 18, machine $M_2$ must contain at least one job $j$ with $g_j = 4$. Let job $\alpha$ be the last job among such jobs. Then, by Lemma 21, we obtain that
\[ (m + 2 - 2\rho_m)L^n > t(S^1_n) + t(S^2_n). \]  
(5)

Since all jobs scheduled after job $\alpha$ on machine $M_2$ are eligible to at most machine $M_3$, we have
\[ 3L^n \geq t(S^1_n) + t(S^2_n) - \left\{ t(S^1_n) + t(S^2_n) \right\}. \]  
(6)

Moreover, by (4) and (6), we have that
\[ t(S^1_n) + t(S^2_n) \geq t(S^1_n) + t(S^2_n) - 3L^n > (2\rho_m - 4)L^n. \]  
(7)

Thus, by (5) and (7), we obtain that
\[ L^n > \frac{2(\rho_m - 2)}{m + 2 - 2\rho_m}L^n. \]

Thus, by Corollary 20, we obtain that
\[ t(S^1_n) + t(S^2_n) > 2(\rho_m - 1)L^n > \frac{4(\rho_m - 1)(\rho_m - 2)}{m + 2 - 2\rho_m}L^n. \]  
(8)

Then, by (4) and (8), we obtain
\[ \sum_{i=1}^{m} t(S^i_n) > \left\{ (2\rho_m - 1) + \frac{4(\rho_m - 1)(\rho_m - 2)}{m + 2 - 2\rho_m} \right\} L^n > mL^n. \]

This is a contradiction.

Case 2: $g_n = 1$

Obviously, machine $M_1$ must contain at least one job $j$ with $g_j \geq 2$. Let job $\beta$ be the last job among jobs that are eligible to machine $M_2$ and are assigned to machine $M_1$. Thus, by Lemma 19, we obtain that
\[ t(S^\beta_1) > (\rho_m - 1)L^\beta. \]  
(9)

All jobs assigned after job $\beta$ to machine $M_1$ are eligible to only machine $M_1$. However, by the minimality, there exists only one such job, job $n$. Since $L^n \geq p_n$, we obtain
\[ t(S^\beta_1) = t(S^\beta_{n-1}) > (\rho_m - 1)L^n. \]  
(10)

We consider two sub-cases concerning the jobs scheduled on machine $M_2$.

Case 2.1: No jobs eligible to machine $M_4$ are assigned to machine $M_2$.

Then, it implies that, by Corollary 18, for any job $j$,
\[ 3L^j \geq t(S^1_j) + t(S^2_j). \]  
(11)

By (9), (10) and $3L^\beta \geq t(S^\beta_1) + t(S^\beta_2)$, we obtain
\[ L^\beta \geq \frac{\rho_m - 1}{4 - \rho_m} L^n. \]

Thus, by (9), we have
\[ t(S^\beta_2) \geq t(S^\beta_2) > (\rho_m - 1)L^\beta > \frac{(\rho_m - 1)^2}{4 - \rho_m}L^n. \]  
(12)

Therefore, by (3) and (12), we obtain that $t(S^1_n) + t(S^2_n) > 3L^n$. But, this is contradictory to (11).

Case 2.2: There exists at least one job with $g_j = 4$ that is assigned to machine $M_2$. Let job $h$ be the last job among such jobs. Thus, by Lemma 21, we have
\[ (m + 2 - 2\rho_m)L^h > t(S^h_1) + t(S^h_2) \geq t(S^h_2). \]  
(13)
We consider job $k$ that is the last job with $g_k = 3$ assigned to machine $M_1$. If there does not exist such a job, $t(S^n_1) \leq 2L^n$ and it is a contradiction. Therefore, job $k$ must exist. Also all jobs scheduled after job $k$ on machine $M_1$ are eligible to at most machines $M_2$. Thus, by Corollary 20, we have

$$t(S^n_k) + t(S^n_3) > 2(\rho_m - 1)t^k.$$  \hfill (14)

Since $2L^n \geq t(S^n_1) - t(S^n_k)$, by (3), we have

$$t(S^n_k) \geq (\rho_m - 2)L^n.$$  \hfill (15)

Now, we consider two cases according to the order of two jobs, jobs $h$ and $k$.

**Case 2.2.1.** Suppose that job $h$ precedes job $k$. Thus, $t(S^n_h) \geq t(S^n_k)$.

Since the jobs scheduled after job $h$ on machine $M_2$ as well as all jobs scheduled on machine $M_1$ are eligible to at most machine $M_2$, we have

$$3L^n \geq t(S^n_1) + t(S^n_2) - t(S^n_k).$$  \hfill (16)

Thus, by (3), (9) and (16), we obtain that

$$t(S^n_2) \geq t(S^n_1) + t(S^n_2) - 3L^n \geq (\rho_m - 3)L^n + (\rho_m - 1)L^\beta.$$  \hfill (17)

Thus, by (13) and (17), we obtain that

$$L^h > \frac{1}{m + 2 - 2\rho_m}t(S^n_h) > \frac{\rho_m - 3}{m + 2 - 2\rho_m}L^n + \frac{\rho_m - 1}{m + 2 - 2\rho_m}L^\beta.$$  

Then, by the above inequality, Lemma 19, Corollary 20, (15) and (1), this implies that

$$t(S^n_h) \geq t(S^n_k) > (\rho_m - 1)L^h \geq \frac{(\rho_m - 1)(\rho_m - 3)}{m + 2 - 2\rho_m}L^n + \frac{(\rho_m - 1)^2}{m + 2 - 2\rho_m}L^\beta.$$  

$$t(S^n_k) \geq 2(\rho_m - 1)L^k$$  

$$t(S^n_h) \geq (\rho_m - 2)L^n$$  

$$mt^k \geq \sum_{i=1}^{m} t(S^n_i).$$

Then, we obtain

$$L^k > \frac{-\rho_m^2 + (m + 2)\rho_m - (2m + 1)}{(m + 2 - 2\rho_m)^2}L^n + \frac{(\rho_m - 1)^2}{(m + 2 - 2\rho_m)^2}L^\beta.$$  

By minimality, job $\beta$ is the job scheduled right before job $n$. Thus, job $\beta$ is scheduled after jobs $h$ and $k$. Thus, by the above inequality, (10), (9), Lemma 19 and (1), we have

$$t(S^n_1) = t(S^n_1 - 1) > (\rho_m - 1)L^n$$  

$$t(S^n_2) \geq (\rho_m - 1)L^\beta$$  

$$t(S^n_3) \geq t(S^n_2) > (\rho_m - 1)t^k$$  

$$t(S^n_4) \geq t(S^n_4) > \frac{(\rho_m - 1)(\rho_m - 3)}{m + 2 - 2\rho_m}L^n + \frac{(\rho_m - 1)^2}{m + 2 - 2\rho_m}L^\beta$$  

$$mt^\beta \geq \sum_{i=1}^{m} t(S^n_i).$$

Then, we obtain

$$L^\beta > \frac{(\rho_m - 1)\left[\rho_m^2 + (2 - 2m)\rho_m + m^2 - m - 3\right]}{(m + 1 - \rho_m)^2(m + 3 - 3\rho_m)}L^n.$$
Thus, by the above inequality, (3), (9) and Lemma 19, we have
\[ t(S_i^n) > \rho_m L^n \]
\[ t(S_2^n) \geq t(S_2^\beta) > (\rho_m - 1)L^\beta \]
\[ > (\rho_m - 1)^2 \left\{ \rho_m^2 + (2 - 2m)\rho_m + m^2 - m - 3 \right\} \frac{L^n}{(m + 1 - \rho_m)^2(m + 3 - 3\rho_m)} \]
\[ t(S_3^n) \geq t(S_3^\beta) \geq t(S_3^h) > (\rho_m - 1)L^k \]
\[ > \frac{(\rho_m - 1)^3}{(m + 2 - 2\rho_m)^2} \left\{ \rho_m^2 + (m + 2)\rho_m - (2m + 1) \right\} + \frac{(\rho_m - 1)^3}{(m + 1 - \rho_m)^2(m + 3 - 3\rho_m)} \] 
\[ L^n \]
\[ t(S_4^n) \geq t(S_4^\beta) \geq t(S_4^h) > (\rho_m - 1)(\rho_m - 3)L^\beta \]
\[ > \frac{(\rho_m - 1)^2}{m + 2 - 2\rho_m} L^n + \frac{(\rho_m - 1)^2}{m + 2 - 2\rho_m} L^\beta \]
\[ > \frac{\rho_m - 1}{m + 2 - 2\rho_m} \left\{ \rho_m^3 + (m + 2)\rho_m - (2m + 1) \right\} \frac{L^n}{(m + 1 - \rho_m)^2(m + 3 - 3\rho_m)} \]

Therefore, we obtain that \[ \sum_{i=1}^{m} t(S_i^n) > mL^n \]. But, it is contradictory to (1).

**Case 2.2.2** Suppose that job \( k \) precedes job \( h \). Thus, \( t(S_i^h) \geq t(S_i^k) \).

We consider the job \( k \). Then, by (14), (15) and (1), we have
\[ t(S_2^k) + t(S_3^k) > 2(\rho_m - 1)L^k \]
\[ t(S_4^k) > (\rho_m - 2)L^h \]
\[ mL^k \geq \sum_{i=1}^{m} t(S_i^k) \]

Then, we obtain
\[ L^k > \frac{\rho_m - 2}{m + 2 - 2\rho_m} L^n \]

Thus, by the above inequality, (15), Lemma 19, Corollary 20 and (1), we have
\[ t(S_1^h) \geq t(S_1^h) > (\rho_m - 2)L^n \]
\[ t(S_2^h) \geq t(S_2^h) > (\rho_m - 1)L^k > \frac{(\rho_m - 1)(\rho_m - 2)}{m + 2 - 2\rho_m} L^n \]
\[ t(S_3^h) + t(S_4^h) > 2(\rho_m - 1)L^h \]
\[ mL^h \geq \sum_{i=1}^{m} t(S_i^h) \]

Then, we have
\[ L^h > \frac{(\rho_m - 2)(m + 1 - \rho_m)}{(m + 2 - 2\rho_m)^2} L^n \]

By minimality, job \( \beta \) is the job scheduled right before job \( n \). Thus, job \( \beta \) is scheduled after jobs \( h \) and \( k \). Then, by (10), (9), Corollary 20 and (1), we have
\[ t(S_1^\beta) = t(S_1^{n-1}) > (\rho_m - 1)L^n \]
\[ t(S_2^\beta) > (\rho_m - 1)L^\beta \]
\[ t(S_3^\beta) + t(S_4^\beta) \geq t(S_3^h) + t(S_4^h) > 2(\rho_m - 1)L^h > \frac{2(\rho_m - 1)(\rho_m - 2)(m + 1 - \rho_m)}{(m + 2 - 2\rho_m)^2} L^n \]
\[ mL^\beta \geq \sum_{i=1}^{m} t(S_i^\beta) \].
Table 3
Overview on the bounds of competitive ratio of the problems.

<table>
<thead>
<tr>
<th>m</th>
<th>Eligibility</th>
<th>LB</th>
<th>UB</th>
<th>Opt</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>General</td>
<td>2.000</td>
<td>2.000</td>
<td>Opt</td>
<td>Azar et al. [4]</td>
</tr>
<tr>
<td>2</td>
<td>GoS</td>
<td>1.667</td>
<td>1.667</td>
<td>Opt</td>
<td>Park et al. [31]</td>
</tr>
<tr>
<td>3</td>
<td>General</td>
<td>2.500</td>
<td>2.500</td>
<td>Opt</td>
<td>Theorem 5</td>
</tr>
<tr>
<td>3</td>
<td>Interval</td>
<td>2.414</td>
<td>↓ 2.500</td>
<td></td>
<td>Lemma 9</td>
</tr>
<tr>
<td>3</td>
<td>Nested</td>
<td>2.333</td>
<td>↓ 2.500</td>
<td></td>
<td>Lemma 12</td>
</tr>
<tr>
<td>4</td>
<td>General</td>
<td>3.000</td>
<td>3.000</td>
<td>Opt</td>
<td>Lim et al. [27]</td>
</tr>
<tr>
<td>4</td>
<td>GoS</td>
<td>2.618</td>
<td>↓ 3.000</td>
<td></td>
<td>Lemma 10</td>
</tr>
<tr>
<td>4</td>
<td>Nested</td>
<td>2.414</td>
<td>↓ 3.000</td>
<td></td>
<td>Lemma 13</td>
</tr>
<tr>
<td>4</td>
<td>GoS</td>
<td>2.000</td>
<td>2.000</td>
<td></td>
<td>Theorem 22</td>
</tr>
<tr>
<td>5</td>
<td>General</td>
<td>3.250</td>
<td>3.250</td>
<td>Opt</td>
<td>Theorem 5</td>
</tr>
<tr>
<td>5</td>
<td>Interval</td>
<td>3.000</td>
<td>↑ 3.250</td>
<td></td>
<td>Lemma 11</td>
</tr>
<tr>
<td>5</td>
<td>Nested</td>
<td>3.000</td>
<td>↑ 3.250</td>
<td></td>
<td>Lemma 14</td>
</tr>
<tr>
<td>5</td>
<td>GoS</td>
<td>2.500</td>
<td>2.510</td>
<td></td>
<td>Theorem 22</td>
</tr>
<tr>
<td>6</td>
<td>General</td>
<td>3.500</td>
<td>3.500</td>
<td>Opt</td>
<td>Theorem 5</td>
</tr>
<tr>
<td>6</td>
<td>Interval</td>
<td>3.000</td>
<td>↑ 3.500</td>
<td></td>
<td>Lemma 15</td>
</tr>
<tr>
<td>6</td>
<td>Nested</td>
<td>3.000</td>
<td>↑ 3.500</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

↓ UB driven by the problem with more general eligibility.
↑ LB driven by the problem with more special eligibility.
.Getenv driven by the problem with the same eligibility and less number of machines.

Fig. 4. Best known lower bounds of the competitive ratio of problems with different number of machines and different eligibility.

Then, we have
\[
L^\beta > \frac{(\rho_m - 1) \left[ 2\rho_m^2 - 2(m + 1)\rho_m + m^2 \right]}{(m + 1 - \rho_m)(m + 2 - 2\rho_m)^2} L_n.
\]

Thus, by the above inequality, (3), (9) and Corollary 20, we obtain that
\[
t(S^g_1) > \rho_m L_n
\]
\[
t(S^g_2) > (\rho_m - 1)L^\beta > \frac{(\rho_m - 1)^2 \left[ 2\rho_m^2 - 2(m + 1)\rho_m + m^2 \right]}{(m + 1 - \rho_m)(m + 2 - 2\rho_m)^2} L_n
\]
\[
t(S^g_3) + t(S^g_4) \geq t(S^g_3) + t(S^g_4) > \frac{2(\rho_m - 1)(\rho_m - 2)(m + 1 - \rho_m)}{(m + 2 - 2\rho_m)^2} L_n.
\]

Therefore, we obtain that \(\sum_{i=1}^m t(S^g_i) > mL_n\). But, it is contradictory to (1). It completes the proof.  \(\square\)
4. Conclusion

Table 3 shows the upper and lower bounds of the optimal competitive ratio for the online scheduling problems with different number of machines and different eligibility environment. Fig. 4 describes the best known lower bounds for different cases in terms of number of machines and eligibility constraints.

As future research, following questions may be considered:

(i) Is algorithm A-W optimal for the online scheduling problem with general eligibility constraints for arbitrary $m$?

(ii) Does algorithm LGF($\rho_m, m$) outperform existing algorithms for the online scheduling with GoS eligibility constraints with a proper parameter $\rho_m$ for $m \geq 6$?

(iii) What is the minimum function of $m$ for $\rho_m$ to guarantee that algorithm LGF($\rho_m, m$) always succeeds in scheduling all jobs with a competitive ratio of $\rho_m$?

Appendix. Supplementary data

Supplementary data associated with this article can be found, in the online version at doi:10.1016/j.tcs.2011.05.029.

References