Coordination mechanisms with hybrid local policies

Kangbok Lee\textsuperscript{a}, Joseph Y-T. Leung\textsuperscript{b,∗}, Michael L. Pinedo\textsuperscript{c}

\textsuperscript{a} Department of Supply Chain Management \& Marketing Sciences, Rutgers Business School, 1 Washington Park, Newark, NJ 07102, USA
\textsuperscript{b} Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA
\textsuperscript{c} Department of Information, Operations \& Management Sciences, Stern School of Business, New York University, 44 West 4th Street, New York, NY 10012-1126, USA

\textbf{A R T I C L E I N F O}

\textbf{Article history:}
Received 9 November 2009
Received in revised form 1 April 2011
Accepted 2 May 2011
Available online 17 June 2011

\textbf{Keywords:}
Coordination mechanism
Price of anarchy
Mixed local policy
Makespan
Total completion time
Eligibility constraints

\textbf{A B S T R A C T}

We study coordination mechanisms for scheduling \(n\) jobs on \(m\) parallel machines where agents representing the jobs interact to generate a schedule. Each agent acts selfishly to minimize the completion time of his/her own job, without considering the overall system performance that is measured by a central objective. The performance deterioration due to the lack of a central coordination, the so-called price of anarchy, is determined by the maximum ratio of the central objective function value of an equilibrium schedule divided by the optimal value. In the first part of the paper, we consider a mixed local policy with some machines using the SPT rule and other machines using the LPT rule. We obtain the exact price of anarchy for the problem of minimizing the makespan and some bounds for the problem of minimizing the total completion time. In the second part of the paper, we consider parallel machine scheduling subject to eligibility constraints. We devise new local policies based on the flexibilities and the processing times of the jobs. We show that the newly devised local policies outperform both the SPT and the LPT rules.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

We consider the distributed scheduling of \(n\) independent jobs on \(m\) parallel machines in which all decisions are interactively done by agents who are in charge of their own jobs. Each agent has a job and selects a machine to process his/her own job so as to maximize his/her own utility, which is defined as the negation of the completion time of his/her own job. Without a central coordination, the jobs assigned to the same machine would compete to finish as early as possible, which will lead to chaos. To resolve the conflict, each machine will announce, in advance, the sequencing rule used by the machine to sequence the jobs assigned to that machine. For example, if the Shortest Processing Time first (SPT) rule is used, jobs assigned to that machine are sequenced in nondecreasing order of their processing times. Similarly, if the Longest Processing Time first (LPT) rule is used, jobs are sequenced in nonincreasing order of their processing times. The sequencing rule used by a machine is usually referred to as the local policy of that machine.

However, there is a central objective function for evaluating the performance of the overall schedule. The central objective may be one of the classical scheduling objectives, such as makespan or total completion time. Due to the lack of a central coordination, the central objective may deteriorate, which is typically referred to as the \textit{Price Of Anarchy} (POA). The POA is closely related to the concept of a Nash equilibrium. A Nash equilibrium schedule is defined as one where no agent can change his/her current decision for a better schedule. In this paper we will only consider pure Nash equilibria which means...
that each agent can only select one machine to process his/her own job. We assume that each agent is well aware of the local policy of each machine as well as all information about other jobs (including their processing times).

The POA can be formally defined as follows. Let $I$ be the set of all instances and $\Lambda(I)$ be the set of all equilibrium schedules of instance $I$. Let $Z$ be the central objective function. Let $\sigma^*$ be an optimal schedule. Then, the POA is defined as

$$\text{POA} = \max_{I \in \Lambda} \max_{\sigma \in \Lambda(I)} \left\{ \frac{Z(\sigma)}{Z(\sigma^*)} \right\}.$$ 

Clearly, the local policy of each machine is a critical factor affecting the overall system performance, and hence the POA. In this paper, we consider the design of local policies in different environments.

### 1.1. Related works

In the literature, different machine environments and different local policies have been considered. Immorlica et al. [1] investigated four different machine environments, where job $j$ has processing time $p_{ij}$ on machine $i$. The four machine environments are: (1) identical machines—$p_{ij} = p_j$ for each job $j$ and machines $i$ and $k$; (2) uniform machines—$p_{ij} = p_j/s_i$, where $p_j$ is the processing requirement of job $j$ and $s_i$ is the speed of machine $i$; (3) parallel machine scheduling with assignment restrictions—job $j$ can only be scheduled on a subset $M_i$ of machines; i.e., $p_{ij} = p_j$ if $i \in M_i$, otherwise, $p_{ij} = \infty$; and (4) unrelated machines—$p_{ij}$ is an arbitrary positive number. As for local policies, they considered SPT, LPT, Randomized and Makespan rules [1].

Most results in the literature are concerned with the makespan as the central objective function. For identical machines, the POA is $2 - \frac{1}{m}$ if the local policy is the SPT rule [2,1], and $\frac{3}{2} - \frac{1}{m}$ if the local policy is the LPT rule [3,4]. For two uniform machines, the POA is $1 + \frac{\sqrt{3}}{2}$ for both the SPT and LPT rules [5]. For three or more uniform machines, the POA is $\Theta(\log m)$ for SPT [1,6], while that of LPT lies within the interval $[1.54, 1 + \frac{\sqrt{3}}{2}]$ [7,8]. For parallel machine scheduling with assignment restrictions, the POA is $\Theta(\log m)$ for both SPT and LPT [9,1]. For unrelated machines, the POA for SPT lies within the interval $[\log m, m]$ [5,10], while that for LPT is unbounded.

Christodoulou et al. [3] consider a hybrid local policy for two identical machines, where the first machine uses the SPT rule and the second machine uses the LPT rule. They show that the POA of this policy has a lower bound of $\frac{7}{4}$. As we shall see in Section 2, we will show that $\frac{7}{4}$ is also an upper bound for this special case.

If the central objective is the total completion time and the machine environment consists of parallel identical machines, the POA is 1 for SPT and $\frac{n}{m}$ for LPT [11]. If the central objective is the total weighted completion time and the machine environment consists of identical machines in parallel, then the POA is $\frac{2}{3}$ for the Weighted Shortest Processing Time first (WSPT) rule [12,13].

### 1.2. Our contribution

In this paper we present new local policies for improving job utility or system performance. The paper consists of two parts: (1) the first part deals with a mixed local policy that will avoid starvation of either long jobs or short jobs, and (2) the second part analyzes a local policy for parallel machine scheduling subject to eligibility constraints so as to improve the system performance.

In the first part we focus on mixed local policies. In the literature, all studies assume that all machines use the same local policy. For example, either all machines use SPT or all machines use LPT. The drawback of this approach is that either the long jobs will be disadvantaged (if SPT is used) or the short jobs will be disadvantaged (if LPT is used). To overcome this drawback, we can consider a policy where some machines use SPT, while other machines use LPT. In this way, the long jobs can choose the machines that use SPT, while the short jobs can choose the machines that use LPT. We call this a Mixed local policy and let $\text{Mixed}(h)$ denote the policy where $h$ machines use SPT and $m - h$ machines use LPT, $1 \leq h \leq m - 1$. We assume that all machines use the job index as a tie-breaking rule just in case two jobs have the same processing time; i.e., the job with the smaller job index will be scheduled before the job with the larger job index. In this paper we analyze the POA of Mixed local policies considering the makespan and the total completion time objectives on identical parallel machines.

In the second part of the paper, we consider parallel machine scheduling subject to eligibility constraints. The set of eligible machines for job $j$, $M_j$, is called the eligible set of job $j$. Previous results have been obtained under the assumption of arbitrary eligible sets [9,1]. However, in practice, eligible sets may have structural properties that depend on the application. We consider two special types of eligibility constraints, namely Grade of Service (GoS) eligibility and Interval (Itrvl) eligibility. For applications of these two eligibilities, the reader is referred to the survey paper by Leung and Li [14]. For GoS eligibility, each job $j$ has a grade $g_j$ and it is only allowed to be processed by machine $i$, where $1 \leq i \leq g_j$. Interval eligibility is a more general version of GoS eligibility in that each job $j$ has a pair of indexes $i_u$ and $i_v$ such that it can be processed by a machine $i$, where $u_i \leq i \leq v_i$. We analyze the POA of traditional local policies such as the SPT and LPT rules. We propose new local policies and show that they outperform the traditional ones.
1.3. Model and notation

Throughout this paper, we use the following notations. There are $n$ jobs to be scheduled on $m$ parallel machines. Job $j$ has processing time $p_j$, $j = 1, \ldots, n$. Let $S_i$ and $S_i^+$ denote the sets of jobs scheduled on machine $i$, $i = 1, \ldots, m$, in the schedule of context and in the optimal schedule, respectively. The starting time and completion time of job $j$ are denoted by $B_j$ and $C_j$, respectively. We consider two types of central objectives: the makespan and the total completion time. For the makespan case, the optimal makespan is denoted by $C_{\text{max}}(\sigma^*)$ and the makespan of schedule $\sigma$ is denoted by $C_{\text{max}}(\sigma)$. For the total completion time case, the optimal total completion time is denoted by $\sum C_j(\sigma^*)$ and the total completion time of schedule $\sigma$ is denoted by $\sum C_j(\sigma)$. For simplicity, we let $P$ denote $\sum_{j=1}^n p_j$.

1.4. Organization of the paper

The paper is organized as follows. In the next section, we focus on Mixed local policies. In Section 2.1, we consider the problem of minimizing the makespan; in Section 2.2, we consider the problem of minimizing the total completion time; and in Section 2.3 we summarize the results. In Section 3, we consider scheduling problems subject to eligibility constraints. In Section 3.1, we consider arbitrary eligibility; in Section 3.2, we consider GoS eligibility; in Section 3.3, we consider interval eligibility; and in Section 3.4 we summarize the results. Finally, we draw some conclusions in Section 4.

2. Mixed local policies

In this section we consider Mixed local policies, with $h$ machines using SPT and $m-h$ machines use LPT to sequence their jobs. We use Mixed$(h)$ to refer to this local policy. Without loss of generality, we may assume that the first $h$ machines use SPT and the last $m-h$ machines use LPT, where $1 \leq h \leq m-1$. We first consider the makespan objective and then the total completion time objective.

For the existence proof of the pure Nash equilibrium, we consider the following algorithm. We assume that jobs are sorted in increasing order of their processing times ($p_1 \leq \cdots \leq p_n$).

Algorithm Mixed $(h)$

\[
\begin{align*}
    &s \leftarrow 1, t \leftarrow 0, S_i \leftarrow \emptyset \quad \text{for } i = 1, \ldots, m. \\
    &\textbf{While} \ (s \leq t) \\
    &\quad i_1 \leftarrow \arg\min_{1 \leq i \leq s} \{\sum_{j \in i} p_j\} \\
    &\quad i_2 \leftarrow \arg\min_{h+1 \leq i \leq m} \{\sum_{j \in i} p_j\} \\
    &\quad \textbf{If} \ (\sum_{j \in S_{i_1}} p_j + p_s \leq \sum_{j \in S_{i_2}} p_j + p_t) \\
    &\quad \quad S_{i_1} \leftarrow S_{i_1} \cup \{s\}, s \leftarrow s + 1 \\
    &\quad \quad \textbf{Else} \ S_{i_2} \leftarrow S_{i_2} \cup \{t\}, t \leftarrow t + 1 \\
\end{align*}
\]

In the resulting schedule there are two groups of machines; on the first $h$ machines, jobs are scheduled in SPT order while on the last $m-h$ machines, jobs are scheduled in LPT order. By [1], LPT and SPT policies in identical parallel machine environments have pure Nash equilibria, respectively. Thus, no job scheduled in either one of the two groups will benefit from a move to another machine in the same group. Moreover, because of the structure of Algorithm Mixed $(h)$, when a job moves to a machine in the other group, no benefit is being obtained either. Thus, the resulting schedule is a pure Nash equilibrium schedule.

2.1. Makespan

For the makespan objective, the POA of Mixed$(h)$ lies between that of the LPT policy and the SPT policy, as the next theorem shows.

**Theorem 1.** The POA of the distributed scheduling problem with $m$ identical machines that follow the Mixed$(h)$ local policy to minimize the makespan is $2 - \min \left\{ \frac{1}{h}, \frac{2}{m+1} \right\}$, $1 \leq h \leq m-1$.

**Proof.** We first establish an upper bound for the POA.

Consider the Nash equilibrium schedule $\sigma$. Let $j$ be the job that finishes last; i.e., $C_{\text{max}}(\sigma) = C_j = B_j + p_j$. In the first $h$ machines, all the time slots before $B_j$ must be occupied by other jobs according to the SPT local policy. When $j$ is scheduled at time $B_j$, all the jobs scheduled before $j$ in the last $m-h$ machines have processing times that are equal or greater than $p_j$, since they are sequenced according to LPT. Thus, $P \geq hB_j + (m-h)p_j + p_j$, and we have

\[
    C_j = B_j + p_j \leq \frac{1}{h}P + \left(1 - \frac{m-h+1}{h}\right)p_j.
\]

We now consider two cases, depending on the sign of $\left(1 - \frac{m-h+1}{h}\right)$. First, we consider the case where $\left(1 - \frac{m-h+1}{h}\right) \geq 0$, or equivalently, $h \leq \frac{m+1}{2}$.
Since $C_{\max}(\sigma^*) \geq \frac{p}{m}$ and $C_{\max}(\sigma^*) \geq p_j$, we have

$$C_{\max}(\sigma) = C_j \leq \frac{mP}{h} + \left(1 - \frac{m-h+1}{h}\right)p_j \leq \left(2 - \frac{1}{h}\right)C_{\max}(\sigma^*).$$

Therefore, the POA has an upper bound

$$\text{POA} \leq \frac{C_{\max}(\sigma)}{C_{\max}(\sigma^*)} \leq 2 - \frac{1}{h}.$$  

Second, we consider the case where $\left(1 - \frac{m-h+1}{h}\right) < 0$, or equivalently $h < \frac{m+1}{2}$.

$$C_{\max}(\sigma) = C_j = \frac{p}{h} - \left(\frac{m-h+1}{h} - 1\right)p_j.$$  

Since the time slots before $B_j$ in the first $m - 1$ machines are occupied by other jobs and the last machine has at least one job which is not job $j$ and has a processing time greater than or equal to $p_j$, $P \geq (m-1)B_j + 2p_j$. Thus, $B_j \leq \frac{p-2p_j}{m-1}$, and we have

$$C_{\max}(\sigma) = C_j = B_j + p_j \leq \frac{1}{m-1}P + \left(1 - \frac{2}{m-1}\right)p_j.$$  

If $p_j > \frac{1}{m+1}P$, then $C_j \leq \frac{p}{h} - \left(\frac{m-h+1}{h} - 1\right)p_j < \frac{2m+1}{m+1}P$ since $\left(1 - \frac{m-h+1}{h}\right) > 0$. If $p_j \leq \frac{1}{m+1}P$, then $C_j \leq \frac{1}{m-1}P + \left(1 - \frac{2}{m-1}\right)p_j \leq \frac{2m}{m+1}P$. Thus, $C_{\max}(\sigma) \leq \frac{2m}{m+1}P$. Since $C_{\max}(\sigma^*) \geq \frac{p}{m}$, we have

$$\text{POA} \leq \frac{C_{\max}(\sigma)}{C_{\max}(\sigma^*)} \leq \frac{p}{m} \leq 2 - \frac{2}{m+1}.$$  

From now on, we proceed with establishing a lower bound for the POA. We consider two cases, depending on whether $h \geq \frac{m+1}{2}$ or not. First, we consider the case where $h \geq \frac{m+1}{2}$. There are $h(h-1)$ jobs with processing time one unit and $m-h+1$ jobs with processing time units. For convenience, let the jobs with processing times 1 (respectively, $h$) be called the small (respectively, big) jobs. Obviously, in an optimal schedule $\sigma^*$ with an optimal makespan of $C_{\max}(\sigma^*) = h$, all small jobs are assigned to the first $h-1$ machines and all big jobs are assigned to the last $m-h+1$ machines. Since the total processing time is $\sum p_j = h(h-1) \times 1 + (m-h+1) \times h = mh$, this schedule is indeed optimal.

In the Nash equilibrium schedule $\sigma$, all small jobs are evenly assigned to machines 1, $\ldots$, $h$ and all big jobs, except the last one, are assigned to machines $h+1$, $\ldots$, $m$. The last big job is assigned to an arbitrary machine among the first $h$ machines. Thus, $C_{\max}(\sigma) = (h-1) + h = 2h-1$. Therefore, the POA has the following lower bound.

$$\text{POA} \geq \frac{C_{\max}(\sigma)}{C_{\max}(\sigma^*)} = \frac{2h-1}{h} = 2 - \frac{1}{h}.$$  

Second, we consider the case where $h < \frac{m+1}{2}$. There are $mh$ jobs with a processing time of one unit and $m-h+1$ jobs with a processing time of $m$ units. For convenience, let the jobs with processing times 1 (respectively, $m$) be called small (respectively, big) jobs. In an optimal schedule $\sigma^*$ with an optimal makespan of $C_{\max}(\sigma^*) = m+1$, each of the first $h-1$ machines has $m+1$ small jobs, while each of the last $m-h+1$ machines has a small job and a big job. Since the total processing time is $\sum p_j = mh \times 1 + (m-h+1) \times m = m(m+1)$, this schedule is indeed optimal.

In the Nash equilibrium schedule $\sigma$, all small jobs are assigned to the first $h$ machines, and all big jobs, except the last one, are assigned to the last $m-h$ machines. The last big job is assigned to an arbitrary machine. Thus, we have $C_{\max}(\sigma) = 2m$. Therefore, the POA has the following lower bound.

$$\text{POA} \geq \frac{C_{\max}(\sigma)}{C_{\max}(\sigma^*)} = \frac{2m}{m+1} = 2 - \frac{2}{m+1}.$$  

From the upper and lower bounds it follows that the POA is exactly $2 - \frac{1}{h}$ for $h \geq \frac{m+1}{2}$ and $2 - \frac{2}{m+1}$ for $h < \frac{m+1}{2}$. \hfill $\Box$

We note that the POA of Mixed$(h)$ lies between that of the SPT case $\left(2 - \frac{1}{m}\right)$ [2.1] and that of the LPT case $\left(\frac{4}{3} - \frac{1}{2m}\right)$ [3]. We summarize the results according to the value of $h$ and $m$ in a graphical way. In Fig. 1, $h = 0$ implies the LPT policy and $h = m$ implies the SPT policy. The $x$-axis represents the ratio of machines using the SPT policy versus the total number of machines $(\frac{h}{m})$ and the $y$-axis represents the POA. As can be seen from Fig. 1, the POA is dependent on $m$ and $h$. Furthermore, as the number of machines increases, there is a tendency that the POA of a mixed local policy is much closer to the POA of the SPT policy even when only one machine follows an SPT policy.
2.2. Total completion time

In this subsection, we consider the total completion time objective. The next theorem gives a lower bound for the POA of this problem.

**Theorem 2.** The POA of the distributed scheduling problem with \( m \) identical machines following the Mixed(\( h \)) local policy to minimize the total completion time is at least \( \frac{m}{h} \).

**Proof.** For a positive number \( l \), let there be \( mlh \) jobs with processing time 1 and \( m - h \) jobs with processing time \( ml \). For convenience, we call the jobs with processing times 1 and processing times \( ml \) the small jobs and big jobs, respectively.

In an optimal schedule \( \sigma^* \), the jobs are scheduled according to the SPT rule [15]: all small jobs are evenly scheduled on all machines and then the big jobs are scheduled later. Thus, we have

\[
\sum C_j(\sigma^*) = \frac{hl(hl + 1)}{2} \times m + (hl + ml)(m - h) = \left( \frac{mh^2}{2} \right) l + \left( \frac{m^2 h^2}{2} \right) l.
\]

In the Nash equilibrium schedule \( \sigma \), all small jobs are evenly scheduled on the first \( h \) machines while the big jobs are scheduled on the last \( m - h \) machines. Thus, we have

\[
\sum C_j(\sigma) = \frac{ml(ml + 1)}{2} \times h + (ml)(m - h) = \left( \frac{m^2 h}{2} \right) l + \left( \frac{m^2 - mh}{2} \right) l.
\]

As \( l \) goes to infinity, the lower bound for the POA approaches \( \frac{m}{h} \). Thus,

\[
\text{POA} \geq \frac{\sum C_j(\sigma)}{\sum C_j(\sigma^*)} \geq \frac{m}{h}. \quad \Box
\]

We now consider the special case where only a single machine uses the SPT rule; i.e., \( h = 1 \). Without loss of generality, we may assume that the jobs are sorted in nondecreasing order. Let \( q_j \) be defined as \( \sum_{k=1}^{j} p_k \). Let \( C_j(\sigma) \) and \( C_j(\sigma^*) \) denote the completion times of job \( j \) in the schedule \( \sigma \) and optimal schedule \( \sigma^* \), respectively.

**Theorem 3.** The POA of the distributed scheduling problem with \( m \) identical machines following the Mixed(1) local policy to minimize the total completion time is exactly \( m \).
Table 1
The POA for mixed local policy.

<table>
<thead>
<tr>
<th>Objective</th>
<th>h</th>
<th>LPT</th>
<th>Mixed (h)</th>
<th>SPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{\text{max}}$</td>
<td>$1 \leq h \leq m - 1$</td>
<td>$\frac{4}{3} - \frac{1}{m}$ [2]</td>
<td>$2 - \min \left{ \frac{1}{h}, \frac{2}{m+1} \right}$</td>
<td>$2 - \frac{1}{m}$ [3, 1]</td>
</tr>
<tr>
<td>$\sum C_j$</td>
<td>$h = 1$</td>
<td>$\frac{7}{2}$ [11]</td>
<td>$m^a$</td>
<td>$1$ [11]</td>
</tr>
<tr>
<td></td>
<td>$2 \leq h \leq m - 1$</td>
<td>$\frac{9}{2}$ [11]</td>
<td>$\geq \frac{m}{\pi}$</td>
<td>$1$ [11]</td>
</tr>
</tbody>
</table>

* Results obtained in this paper.

**Proof.** We consider a Nash equilibrium schedule $\sigma$. Let there be $k$ jobs scheduled on the first machine. Then, we have

$$\sum_{j \in S_1} C_j(\sigma) \leq \sum_{j \in S_1} q_j.$$ 

Since $\sigma$ is a Nash equilibrium schedule, for $j \in S_i$, $i = 2, \ldots, m$, $C_j(\sigma) \leq q_i$. Otherwise, job $j$ should be scheduled on machine 1. Therefore, we have

$$\sum_{j = 1}^n C_j(\sigma) \leq \sum_{j = 1}^n q_j.$$ 

In an optimal schedule $\sigma^*$, we may assume that all jobs are scheduled according to SPT [15]. Thus, we have

$$\sum_{j = 1}^n C_j(\sigma^*) \geq \sum_{j = 1}^n \frac{1}{m} q_j.$$ 

It follows that

$$\text{POA} \leq \frac{\sum C_j(\sigma)}{\sum C_j(\sigma^*)} \leq m.$$ 

By Theorem 2, when $h = 1$, the POA is at least $m$. Therefore, the POA of this problem is exactly $m$. □

2.3. Summary

We summarize the POA of the distributed scheduling problem on $m$ identical machines with Mixed$(h)$ local policy in Table 1.

3. Local policies for eligibility constraints

In this section, we consider parallel machine scheduling subject to eligibility constraints. The central objective is the makespan. We deal with arbitrary eligibility, GoS eligibility and Interval eligibility.

3.1. Arbitrary eligibility

For arbitrary eligibility, Immorlica et al. [1] have shown that the POA for the SPT and LPT rules is $\Theta(\log m)$ based on the analysis of the list scheduling algorithm by Azar et al. [9]. Azar et al. [9] obtained a lower and upper bound for the list scheduling algorithm of $\lceil \log_2 (m + 1) \rceil$ and $\lceil \log_2 m \rceil + 1$, respectively. Hwang et al. [16] provided a slightly sharper upper bound of $\log_2 \frac{4}{3} m - \frac{1}{\lambda}$, where $\lambda$ is the cardinality of the eligible set of the job determining the makespan. Thus, by combining these results, we have the following theorem.

**Theorem 4** ([9, 16]). The POA of the distributed scheduling problem with eligibility constraints and $m$ machines that follow either the SPT or the LPT local policy to minimize the makespan lies within the interval $[\lceil \log_2 (m + 1) \rceil, \log_2 m + 1]$.

We now consider a new local policy, the Least Flexible Job first (LFJ) rule, which is derived from the corresponding list scheduling algorithm. The LFJ list scheduling algorithm sorts the jobs in nondecreasing order of the cardinality of their eligible sets; i.e., job $j$ precedes job $k$ if $|M_j| < |M_k|$. The next job $j$ is assigned to the machine $i \in M_j$ which finishes the earliest. Ties can be resolved in an arbitrary manner. We now adapt the LFJ list scheduling algorithm to be the LFJ local policy. The machine that uses the LFJ local policy will process jobs in nondecreasing order of the cardinality of their eligible sets; ties are resolved by processing the job with the smaller job index first. (We assume that each agent chooses an eligible machine for his/her own job.) As it turns out, the POA of the LFJ rule is slightly better than that of the SPT and LPT rules, as the following theorem shows.
Theorem 5. The POA of the distributed scheduling problem on two identical machines with eligibility constraints and LFJ local policy to minimize the makespan is $\frac{3}{2}$.

Proof. We first obtain an upper bound for the POA.

Let $j$ be the job that finishes last in a Nash equilibrium schedule $\sigma$. If $|M_j| = 1$, then all jobs scheduled before $j$ on the same machine as $j$ are eligible only for that machine. This implies that $C_{\text{max}}(\sigma) = C_{\text{max}}(\sigma^*)$. If $M_j = \{1, 2\}$, then all time slots on both machines before the starting time of $j$, $B_j$, must be occupied by other jobs. Therefore, $P \geq 2B_j + p_j$; i.e. $B_j \leq \frac{1}{2}(P - p_j)$.

Therefore, we have

$$C_{\text{max}}(\sigma) = C_j = B_j + p_j \leq \frac{1}{2}P + \frac{1}{2}p_j.$$ 

Since $C_{\text{max}}(\sigma^*) \geq \frac{1}{2}P$ and $C_{\text{max}}(\sigma^*) \geq p_j$, we have

$$C_{\text{max}}(\sigma) \leq C_{\text{max}}(\sigma^*) + \frac{1}{2}C_{\text{max}}(\sigma^*) = \frac{3}{2}C_{\text{max}}(\sigma^*).$$

Thus, the POA is at most $\frac{3}{2}$.

We proceed with determining a lower bound for the POA.

We consider the following instance. There are three jobs with $p_1 = p_2 = \frac{1}{2}$ and $p_3 = 1$, and $M_1 = M_2 = M_3 = \{1, 2\}$. In an optimal schedule, $C_{\text{max}}(\sigma^*) = 1$ by setting $S_1^1 = \{1, 2\}$ and $S_2^1 = \{3\}$, while in a Nash equilibrium schedule $\sigma$, $C_{\text{max}}(\sigma) = \frac{3}{2}$ by setting $S_1 = \{1, 3\}$ and $S_2 = \{2\}$. Thus, the POA is at least $\frac{3}{2}$. $\square$

Theorem 6. The POA of the distributed scheduling problem on $m$ identical machines with eligibility constraints and LFJ local policy to minimize the makespan lies within the interval $\left[\lceil \log_2(m + 1) \rceil - 1, \log_2 m + \frac{1}{2} \right]$ for $m \geq 3$.

Proof. We first obtain an upper bound for the POA.

By almost the same argument as Immorlica et al. [1], we can show that the set of Nash equilibrium schedules for the LFJ local policy is precisely the set of schedules that can be obtained by the LFJ list scheduling algorithm. Due to a result by Hwang et al. [16], the POA has an upper bound of $\log_2 \frac{4}{3}m - \frac{1}{2}$, where $m$ is the number of machines eligible for processing for the job with the last completion time.

Let $j$ be the job that finishes last in a Nash equilibrium schedule $\sigma$. If $|M_j| = 1$, then all jobs scheduled before $j$ on the same machine as $j$ are eligible only for that machine. This implies that $\sigma$ is an optimal schedule. If $|M_j| \geq 2$, then the POA is upper bounded by $\log_2 \frac{4}{3}m - \frac{1}{2} = \log_2 m + \frac{1}{2}$. Therefore, the POA has an upper bound of $\log_2 m + \frac{1}{2}$.

We proceed with determining a lower bound for the POA.

We consider the problem instance where $2^k \leq m < 2^{k+1}$ and $n = 2^k$ for some positive integer $k$. Every job has one unit of processing time. There are $k + 1$ groups of jobs. Let job $(h, j)$ denote the $j$-th job in group $h$, $h = 1, \ldots, k + 1$. Group $h$, $h = 1, \ldots, k$, consists of $2^{k-h}$ jobs and job $(h, j)$ is eligible for the two machines $h$ and $j + 2^{k-h}$. Group $k + 1$ consists of one job that is eligible for machines $1$ and $2$. Note that all jobs are eligible for two machines. In an optimal schedule $\sigma^*$, each job in group $h$, $h = 1, \ldots, k$, selects the machine with the larger index in its eligible set and the job in group $k + 1$ selects machine $1$. Thus, $C_{\text{max}}(\sigma^*) = 1$. In a Nash equilibrium schedule, $\sigma$, jobs are scheduled in increasing order of group index and each job in group $h$, $h = 1, \ldots, k$, selects the machine with the smaller index in its eligible set. Finally, the job in group $k + 1$ selects machine $2$. Thus, $C_{\text{max}}(\sigma) = k = \lceil \log_2(m + 1) \rceil - 1$. Therefore, the POA is at least $\lceil \log_2(m + 1) \rceil - 1$.

From the upper and lower bounds it follows that the POA is exactly $\frac{3}{2}$ for $m = 2$ and lies within interval $\left[\lceil \log_2(m + 1) \rceil - 1, \log_2 m + \frac{1}{2} \right]$ for $m \geq 3$. $\square$

When the flexibility is utilized in the local policy, the POA is slightly improved for the scheduling problem subject to arbitrary eligibility constraints. However, when it is used for the case with more structured eligibility constraints, the POA can be significantly improved, as we show in the following two subsections. Note that Immorlica et al. [1] have shown that any deterministic local policy for the problem subject to arbitrary eligibility constraints has a POA of $\Theta(\log m)$.

3.2. GoS eligibility

In this subsection, we consider the scheduling of jobs subject to GoS eligibility constraints. Hwang et al. [17] have given examples showing that the POA of the SPT and LPT rules is at least $\lceil \log_2(m + 1) \rceil$. Thus, we have the following theorem.

Theorem 7 ([17]). The POA of the distributed scheduling problem with GoS eligibility constraints and $m$ machines that follow either the SPT or the LPT local policy to minimize the makespan is at least $\lceil \log_2(m + 1) \rceil$.

In a search for a better local policy, we consider one that is adapted from a list scheduling algorithm, which we refer to as the *Lowest Grade and Longest Processing Time first* (LG–LPT) algorithm. The LG–LPT algorithm works as follows. Initially, jobs are sorted such that job $j$ precedes job $k$

- if $g_j < g_k$ or
- if $g_j = g_k$ and $p_j > p_k$ or
- if $g_j = g_k$ and $p_j = p_k$ and $j < k$.
After the sorting, jobs are scheduled in this order. The next job \( j \) is assigned to a machine \( i \in M_j \) that finishes the earliest. Hwang et al. [17] have shown that the LG–LPT algorithm has a tight worst-case performance ratio of 5/4 for \( m = 2 \) and \( 2 - \frac{1}{m-1} \) for \( m \geq 3 \).

We adapt the LG–LPT algorithm as a local policy. A machine that uses the LG–LPT local policy schedules job \( j \) before job \( k \)

- if \( g_j < g_k \) or
- if \( g_j = g_k \) and \( p_j > p_k \) or
- if \( g_j = g_k \) and \( p_j = p_k \) and \( j < k \).

In the next theorem we will show that the LG–LPT local policy has a POA that is the same as the worst-case performance ratio of the LG–LPT algorithm. This is a great improvement over SPT and LPT which have a POA of at least \([\log_2(m + 1)]\).

**Theorem 8.** The POA of the distributed scheduling problem with eligibility constraints and two machines that follow the LG–LPT local policy to minimize the makespan is 5/4.

**Proof.** We first establish an upper bound for the POA.

Assume job \( j \) is the job that finishes last in a Nash equilibrium schedule \( \sigma \). If \( g_j = 1 \), then all jobs scheduled on machine 1 must be scheduled on machine 1, which implies that the current schedule is optimal. Therefore, we may assume that \( g_j = 2 \).

Since \( \sigma \) is a Nash equilibrium schedule, if job \( j \) is scheduled on machine 1, then \( C_j = (\sum_{k \in S_1(\sigma)} p_k + p_j) \) and \( C_j = (\sum_{k \in S_2(\sigma)} p_k + p_j) \). If job \( j \) is scheduled on machine 2, then \( C_j \leq (\sum_{k \in S_1(\sigma)} p_k + p_j) \) and \( C_j = (\sum_{k \in S_2(\sigma)} p_k + p_j) \). So, in both cases, we have

\[
2C_{\text{max}}(\sigma) \leq (\sum_{k \in S_1(\sigma)} p_k + p_j) + (\sum_{k \in S_2(\sigma)} p_k + p_j) = P + p_j.
\]

Since \( C_{\text{max}}(\sigma^*) \geq \frac{5}{4} P \), we have

\[
C_{\text{max}}(\sigma) \leq \frac{1}{2} P + \frac{1}{2} p_j \leq C_{\text{max}}(\sigma^*) + \frac{1}{2} p_j.
\]

There are two cases to consider, depending on the value of \( p_j \). If \( p_j \leq \frac{1}{2} C_{\text{max}}(\sigma^*) \), then we have

\[
C_{\text{max}}(\sigma) \leq \left(1 + \frac{1}{4}\right) C_{\text{max}}(\sigma^*) = \frac{5}{4} C_{\text{max}}(\sigma^*)
\]

On the other hand, if \( p_j > \frac{1}{2} C_{\text{max}}(\sigma^*) \), then we assert that \( C_{\text{max}}(\sigma) = C_{\text{max}}(\sigma^*) \). First, we observe that the number of jobs that start before job \( j \) and are eligible for machine 2 cannot be more than one, since \( p_j > \frac{1}{2} C_{\text{max}}(\sigma^*) \). Therefore, we may assume that there is exactly one such job, denoted by \( l \). Job \( l \) is scheduled on machine 2 and \( p_l \geq p_j > \frac{1}{2} C_{\text{max}}(\sigma^*) \). Therefore, in an optimal schedule \( \sigma^* \), job \( j \) cannot be scheduled on the same machine. If \( \sum_{k \in S_1(\sigma)} p_k < p_j \), then job \( j \) is scheduled on machine 1 in \( \sigma \), just like an optimal schedule \( \sigma^* \). On the other hand, if \( \sum_{k \in S_1(\sigma)} p_k \geq p_j \), then it is impossible to have a schedule with makespan \( C_{\text{max}}(\sigma^*) \). This contradicts the fact that there is an optimal schedule with makespan \( C_{\text{max}}(\sigma^*) \).

In both cases, \( C_{\text{max}}(\sigma) \leq \frac{5}{4} C_{\text{max}}(\sigma^*) \). Therefore, when \( m = 2 \), the POA has an upper bound of \( \frac{5}{4} \).

We now proceed with establishing a lower bound for the POA.

Consider an instance where there are four jobs such that \( g_1 = 1, g_2 = g_3 = g_4 = 2, p_1 = 1/4, p_2 = 3/4 \) and \( p_3 = p_4 = 1/2 \). The optimal makespan is 1 by setting \( S_1^* = \{1, 2\} \) and \( S_2^* = \{3, 4\} \). In a Nash equilibrium schedule \( \sigma, S_1 = \{1, 3, 4\} \) and \( S_2 = \{2\} \). The makespan of \( \sigma \) is 5/4. Therefore, the POA is at least \( \frac{5}{4} \) for \( m = 2 \). □

**Theorem 9.** The POA of the distributed scheduling problem with eligibility constraints and \( m \) machines that follow the LG–LPT local policy to minimize the makespan is \( 2 - \frac{1}{m-1} \) for \( m \geq 3 \).

**Proof.** We first establish an upper bound for the POA.

Assume again that job \( j \) is the one that finishes last in a Nash equilibrium schedule \( \sigma \) and let \( g_j = i \). If \( i = 1 \), then all jobs scheduled on machine 1 must be scheduled on machine 1. Therefore, \( \sigma \) is an optimal schedule already. Thus, we may assume that \( i \geq 2 \).

All the time slots before the starting time of job \( j \), \( B_j \), on machines 1, \ldots, \( i \) must be occupied by other jobs. Note that machine \( i \) has at least one job; otherwise, job \( j \) would be moved to machine \( i \). Consider the jobs scheduled before \( B_j \) on machine \( i \). Since they are scheduled on machine \( i \), their grades are greater than or equal to \( i \). Also, since they are scheduled before \( B_j \), their grades are smaller than or equal to \( i \). Therefore, their processing times are greater than or equal to \( p_j \) by the LG–LPT local policy. Thus, the total processing times of the jobs whose grades are at most \( i \) is \( \sum_{k \leq i} p_k \geq (i - 1)B_j + 2p_j \).

Thus, we have

\[
C_{\text{max}}(\sigma) = B_j + p_j \leq \frac{\sum_{k \leq i} p_k - 2p_j}{i - 1} + p_j \leq \frac{1}{i - 1} \sum_{k \leq i} p_k + \left(1 - \frac{2}{i - 1}\right)p_j.
\]
Since \( C_{\text{max}}(\sigma^*) \geq \frac{1}{i} \sum_{k: \sigma_k, \sigma_{k+1} \in p_k} p_k \) and \( C_{\text{max}}(\sigma^*) \geq p_j \), we have
\[
C_{\text{max}}(\sigma) \leq \frac{i}{i-1} C_{\text{max}}(\sigma^*) + \left( 1 - \frac{2}{i-1} \right) C_{\text{max}}(\sigma^*) \leq \left( 2 - \frac{1}{m-1} \right) C_{\text{max}}(\sigma^*).
\]
Therefore, an upper bound for the POA when \( m \geq 3 \) is \( 2 - \frac{1}{m-1} \).

We now proceed with establishing a lower bound for the POA.

Consider an instance where there are three groups of jobs. In the first group, there are \( m - 2 \) jobs with processing time \( 1 - \frac{1}{m-1} \) and grade \( m - 1 \). In the second group, there are \( m - 2 \) jobs with processing time \( \frac{1}{m-1} \) and grade \( m - 1 \). In the third group, there are two jobs with processing time 1 and grade \( m \). Therefore, under the LG–LPT local policy, the jobs in the first group have higher priority than the jobs in the second group, and the jobs in the second group have higher priority than the jobs in the third group.

In an optimal schedule \( \sigma^* \), one job from the first group and one job from the second group are scheduled on machine \( i \), \( i = 1, \ldots, m - 2 \). On machines \( m - 1 \) and \( m \), jobs from the third group are scheduled, one job per machine. Thus, the optimal makespan is \( C_{\text{max}}(\sigma^*) = 1 \).

In a Nash equilibrium schedule \( \sigma \), all jobs in the first group are scheduled on machines 1, \ldots, \( m - 2 \), one job per machine. All jobs in the second group are scheduled on machine \( m - 1 \). Finally, one job from the third group is scheduled on machine \( m \) and the other job is scheduled on machine 1. Thus, the makespan of \( \sigma \) is \( C_{\text{max}}(\sigma) = 2 - \frac{1}{m-1} \). Therefore, the POA is at least \( 2 - \frac{1}{m-1} \) for \( m \geq 3 \). 

### 3.3. Interval eligibility

In the Interval eligibility case, job \( j \) has two indexes, \( u_j \) and \( v_j \), such that \( M_j = \{ i \mid u_j \leq i \leq v_j \} \). Since GoS eligibility is a special case of Interval eligibility, the POA of the SPT and LPT local policy is at least \( \lceil \log_2 m + 1 \rceil \). In a search for a better local policy, we consider adapting a list scheduling algorithm called the Least Flexible and Longest Processing Time first (LF–LPT) algorithm. In the LF–LPT algorithm, jobs are sorted in nondecreasing order of their flexibility (which is simply the cardinality of the eligible set of the job), and in case of a tie, in nonincreasing order of their processing times. After the jobs have been sorted, the next job \( j \) is assigned to the machine \( i \in M_j \) that finishes first. Lee \[18\] has shown that the LF–LPT algorithm has a worst-case performance ratio is in the interval \( [4 - \frac{2}{\lambda}, 4 - \frac{3}{\lambda}] \) for \( \lambda \geq 4 \), where \( \lambda \) is the cardinality of the eligible set of the most flexible job.

We adapt the LF–LPT rule as a local policy. Each machine schedules the jobs in nondecreasing order of the flexibility of the job, and in case of a tie, in nonincreasing order of their processing times. Further ties will be broken by the job index; i.e., a job with a smaller job index will be scheduled before a job with a larger job index. In the next two theorems, we will establish a value for the POA of the LF–LPT local policy by utilizing the worst-case performance ratio of the LF–LPT list scheduling algorithm.

**Theorem 10.** The POA of the distributed scheduling problem with Interval eligibility constraints and two machines that follow the LF–LPT local policy to minimize the makespan is \( \frac{5}{4} \).

**Proof.** We first establish an upper bound for the POA. Let \( j \) be the job that finishes last in a Nash equilibrium schedule \( \sigma \). If \( j \) is eligible for only one machine, then \( C_{\text{max}}(\sigma) = C_{\text{max}}(\sigma^*) \). Henceforth, we consider the case where \( j \) is eligible for both machines.

Without loss of generality, we may assume that \( j \) is scheduled on machine 1. Let \( L_1 \) and \( L_2 \) be the total processing time of jobs whose eligible sets are \{1\} and \{2\}, respectively. We now consider jobs that are eligible for both machines and start before \( j \). If there are no such jobs, then \( \sigma \) is already optimal and \( C_{\text{max}}(\sigma) = C_{\text{max}}(\sigma^*) \). If there is such a job, let \( k \) be the job. Since \( k \) starts before \( j \), we have \( p_k \geq p_j \). We consider two cases depending on the value of \( p_j \).

If \( p_j > \frac{1}{2} C_{\text{max}}(\sigma^*) \), then we have \( p_k \geq p_j > \frac{1}{2} C_{\text{max}}(\sigma^*) \). This implies that \( k \) is the only job eligible for both machines and starting before \( j \). Now, if \( k \) is scheduled on machine 1, then \( L_2 \geq p_k > \frac{1}{2} C_{\text{max}}(\sigma^*) \). Otherwise, job \( j \) would not be scheduled on machine 1. Thus, it is impossible to have a schedule with a makespan equal to \( C_{\text{max}}(\sigma^*) \). Therefore, \( k \) must be scheduled on machine 2, which implies that \( L_1 \geq L_2 \). Hence, \( j \) is scheduled on machine 1, \( k \) is scheduled on machine 2, and \( L_1 \geq L_2 \) in \( \sigma \). It is easy to see that \( \sigma \) is in fact an optimal schedule.

We now consider the case where \( p_j \leq \frac{1}{2} C_{\text{max}}(\sigma^*) \). Since \( \sigma \) is a Nash equilibrium schedule, all time slots before \( B_j \) are occupied by other jobs. Thus, \( P \geq 2B_j + p_j \), where \( P \) is the total processing time of all the jobs. Thus, \( B_j \leq \frac{1}{2}(P - p_j) \). Since \( C_{\text{max}}(\sigma^*) \geq \frac{1}{2} P \) and \( C_{\text{max}}(\sigma^*) \geq p_j \), we have
\[
C_{\text{max}}(\sigma) = C_j = B_j + p_j \leq \frac{1}{2}(P + p_j) \leq \frac{5}{4} C_{\text{max}}(\sigma^*).
\]
Therefore, the POA has an upper bound of \( \frac{5}{4} \).
Since when \( m = 2 \), the lower bound example for the GoS case will work in this case as well, the POA for the problem is exactly \( \frac{5}{4} \). □

**Theorem 11.** The POA of the distributed scheduling problem with Interval eligibility constraints and \( m \) machines that follow the LF–LPT local policy to minimize the makespan is no more than \( 4 - \frac{3}{m} \) for \( m \geq 3 \).

**Proof.** Let \( j \) be the job that finishes last in a Nash equilibrium schedule \( \sigma \), and let \( \lambda \) be the cardinality of the eligible set of \( j \), i.e. \( \lambda = |M_j| = v_j - u_j + 1 \).

Since \( \sigma \) is a Nash equilibrium schedule, all time slots before the starting time of \( j \), \( B_j \), are occupied by jobs with higher priority than \( j \) under the LF-LPT local policy. We consider the set of jobs that can be scheduled on one of the machines \( u_j, \ldots, v_j \). Each of these jobs must have the cardinality of their eligible sets bounded above by \( \lambda \). Let \( j' \) be a super set of the set of such jobs defined as follows.

\[
J' = \{ k \mid u_k \geq u_j - \lambda + 1 \text{ and } v_k \leq v_j + \lambda - 1 \}.
\]

Since the local policy is LF-LPT, all jobs occupying the time slots before \( B_j \) on machines \( u_j, \ldots, v_j \) must belong to \( J' \). Clearly, \( j \) belongs to \( J' \) as well. Thus, \( \sum_{k \in J'} p_k \geq \lambda B_j + p_j \). Therefore, we have

\[
B_j \leq \frac{1}{\lambda} \left( \sum_{k \in J'} p_k - p_j \right).
\]

Also, all jobs in \( J' \) must be scheduled on machines \( u_j - \lambda + 1, \ldots, v_j + \lambda - 1 \). Thus, \( C_{\max}(\sigma^*) \geq \frac{1}{\lambda} \sum_{k \in J'} p_k \). Note that \( C_{\max}(\sigma^*) \geq p_j \) as well. Therefore, we have

\[
C_{\max}(\sigma) = C_j = B_j + p_j \leq \frac{1}{\lambda} \sum_{k \in J'} p_k + \left( 1 - \frac{1}{\lambda} \right) p_j \leq \left( \frac{3\lambda - 2}{\lambda} \right) C_{\max}(\sigma^*) + \left( 1 - \frac{1}{\lambda} \right) C_{\max}(\sigma^*) = \left( 4 - \frac{3}{\lambda} \right) C_{\max}(\sigma^*).
\]

Since \( \lambda \leq m \), the POA is at most \( 4 - \frac{3}{m} \). □

**Theorem 12.** The POA of the distributed scheduling problem with Interval eligibility constraints and \( m \) machines that follow the LF–LPT local policy to minimize the makespan is at least \( 4 - \frac{28}{m+12} \) for \( m \) being a multiple of 8.

**Proof.** Consider the following instance: there are six groups of jobs to be scheduled on \( m \) machines. We assume that \( m \) is a multiple of eight and let \( q \) be \( \frac{m}{2} \). The information about the jobs are as follows. A group may consist of subgroups and a subgroup consists of jobs. Therefore, each job is denoted by a triplet \((a, b, k)\), where \( a \) is the group index, \( b \) is the subgroup index in the group and \( k \) is the job index in the subgroup. The jobs information are summarized as follows.

| Gr. | S.Gr. | Job | Range | \(|M_j|\) | \(u_j\) | \(v_j\) | \(p_j\) |
|-----|-------|-----|-------|-------|-------|-------|-------|
| 1   | 1     | (1, 1, k) | \(k = 1, \ldots, q - 1\) | 1 | \(q + k\) | \(q + k\) | \(\frac{q-k}{q}\) |
| 2   | 2     | (2, 1, k) | \(k = 1, \ldots, q\) | 1 | \(3q - k + 1\) | \(3q - k + 1\) | \(\frac{q-k}{q}\) |
| 3   | 1     | (3, 1, k) | \(k = 1, \ldots, q\) | 1 | \(k\) | \(k\) | \(\frac{q-k}{q}\) |
| 4   | 2     | (2, 2, k) | \(k = 1, \ldots, q\) | 1 | \(3q + k\) | \(3q + k\) | \(\frac{q-k}{q}\) |
| 5   | 1     | (5, 1, k) | \(k = 1, \ldots, \left(\frac{q}{2} + 1\right)\) | 1 | \(2q + k\) | \(3q + k\) | \(1 - \frac{q-k}{q}\) |
| 6   | 1     | (6, 1, k) | \(k = 1, \ldots, \left(\frac{q}{2} + 1\right)\) | 1 | \(2q + \frac{k}{2}\) | \(2q + \frac{k}{2}\) | \(\frac{q-k}{q+3}\) |

An illustrative example is presented in Fig. 2, where \( m = 32 \).

In an optimal schedule \( \sigma^* \), all jobs in groups 1 and 2 are scheduled on their only eligible machine. Job \((3, 1, k)\) is scheduled on machine \( k \) and job \((3, 2, k)\) is scheduled on machine \( 3q + k \). Job \((4, 1, k)\) is scheduled on machine \( q + k \) and job \((4, 2, k)\) is scheduled on machine \( 3q + 1 - k \). Jobs in subgroup \((5, 1)\) are scheduled on machines from \( q + \frac{k}{2} \) to \( 2q \) evenly and jobs in subgroup \((5, 2)\) are scheduled on machines from \( 2q + 2 \) to \( 2q + \frac{k}{2} \) evenly. The last job in group 6 is scheduled on machine \( 2q + 1 \). Thus, the optimal makespan is 1.
Fig. 2. An illustrative example for a lower bound of POA.

The Nash equilibrium schedule $\sigma$ is constructed by the LF–LPT list scheduling algorithm. All jobs in groups 1 and 2 are scheduled on their only eligible machine. Job $(3, 1, k)$ is scheduled on machine $q+k$ and job $(3, 2, k)$ is scheduled on machine $3q+k$. Jobs in group 4 are scheduled on machines from $2q - \frac{q}{2} + 1$ to $2q + \frac{q}{2} - 2$. Then, jobs in group 5 are scheduled on machines from $q + \frac{q}{2}$ to $2q + \frac{q}{2} + 2$. Finally, the job in group 6 is scheduled on machine $2q + 1$. Thus, the makespan of $\sigma$ is $(1 - \epsilon) + \frac{2q-1}{q+3}(1 - \epsilon) + 1$. As $\epsilon$ goes to zero, the makespan approaches $2 + \frac{2q-1}{q+3}$.

Since $m = 4q$, the POA has the following lower bound.

$$\text{POA} \geq 2 + \frac{2q-1}{q+3} = 4 - \frac{28}{m+12}.$$  

3.4. Summary

We summarize the results of this section in Table 2.

4. Concluding remarks

In the first part of the paper, we analyze the POA under a new local policy, with some machines using SPT as their local policy and with other machines using LPT as their local policy. For the problem of minimizing the total completion time, an upper bound for the POA has remained elusive. The POA in different machine environments, such as uniform machines, is an unexplored area.

In the second part of the paper, we consider the problem with eligibility constraints and develop local policies that are based on the flexibilities as well as on the processing times of the jobs. One possible extension is to develop a local policy with a better POA value. Another possible extension is an answer to the question whether there is a lower bound for the POA for an arbitrary local policy. For the arbitrary eligibility case, there is a lower bound for the POA, $\Omega(\log m)$, under an arbitrary local policy. Similarly, whether there exists for the problem subject to GoS or Interval eligibility constraints with arbitrary local policies a lower bound for the POA also remains an interesting open problem.

Immonen et al. [1] proved that the parallel machine scheduling problem with eligibility constraints and SPT or LPT as the local policy has a pure Nash equilibrium. However, for the local policy utilizing eligibility information, the existence of pure Nash equilibrium has not yet been shown, which is an interesting research question.
Table 2
The POA for models subject to eligibility constraints.

<table>
<thead>
<tr>
<th>Eligibility</th>
<th>Local policy</th>
<th>m</th>
<th>POA</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary</td>
<td>SPT</td>
<td>2</td>
<td>2</td>
<td>[9, 1]</td>
</tr>
<tr>
<td></td>
<td>LPT</td>
<td>m</td>
<td>⌈log_2(m + 1)⌉, log_2(m + 1)</td>
<td>[9, 1, 16]</td>
</tr>
<tr>
<td></td>
<td>LFJ</td>
<td>2</td>
<td>2</td>
<td>[9, 1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m</td>
<td>⌈log_2(m + 1)⌉, log_2(m + 1)</td>
<td>[9, 1, 16]</td>
</tr>
<tr>
<td>GoS</td>
<td>SPT</td>
<td>2</td>
<td>2</td>
<td>[17] a</td>
</tr>
<tr>
<td></td>
<td>LPT</td>
<td>m</td>
<td>Θ(log m)</td>
<td>[17] a</td>
</tr>
<tr>
<td></td>
<td>LG–LPT</td>
<td>2</td>
<td>5/4</td>
<td>[17] a</td>
</tr>
<tr>
<td></td>
<td>LF–LPT</td>
<td>m</td>
<td>2 − 1 (\frac{1}{m−1})</td>
<td>[17] a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m</td>
<td>⌈log_2(m + 1)⌉ − 1, log_2(m + 1)</td>
<td>[17] a</td>
</tr>
</tbody>
</table>

a Result obtained in this paper.

Acknowledgments

The second author’s work was supported in part by the NSF Grant CMMI-0969830. The third author’s work was supported in part by the NSF Grant CMMI-0969755.

References