4OR-Q J Oper Res (2010) 8:331–364
DOI 10.1007/s10288-010-0149-1

INVITED SURVEY

Makespan minimization in online scheduling with machine eligibility

Kangbok Lee · Joseph Y.-T. Leung · Michael L. Pinedo

Received: 18 August 2010 / Revised: 5 November 2010 / Published online: 23 November 2010
© Springer-Verlag 2010

Abstract In this paper we provide a survey of online scheduling in parallel machine environments with machine eligibility constraints and the makespan as objective function. We first give a brief overview of the different parallel machine environments and then survey the various types of machine eligibility constraints, including tree-hierarchical processing sets, Grade of Service processing sets, interval processing sets, and nested processing sets. We furthermore describe the relationships between the various different types of processing sets. We proceed with describing two basic online scheduling paradigms, namely online over list and online over time. For each one of the two paradigms we survey all the results that have been recorded in the literature with regard to each type of machine eligibility constraints. We obtain also

This work is supported in part by the NSF Grant CMMI-0969830. M. L. Pinedo is supported in part by the NSF Grant CMMI-0969755.

K. Lee
Department of Supply Chain Management and Marketing Sciences,
Rutgers Business School, 1 Washington Park,
Newark, NJ 07102-3122, USA
e-mail: kangblee@business.rutgers.edu

J. Y.-T. Leung
Department of Computer Science, New Jersey Institute of Technology,
Newark, NJ 07102, USA
e-mail: leung@cis.njit.edu

M. L. Pinedo (✉)
Department of Information, Operations and Management Sciences,
Stern School of Business,
New York University, 44 West 4th Street,
New York, NY 10012-1126, USA
e-mail: mpinedo@stern.nyu.edu

several extensions in various directions. In the concluding section we describe the most important open problems in this particular area.

Keywords Parallel machine scheduling · Eligibility constraint · Tree-hierarchical and GoS processing sets · Interval and nested processing sets · Online and Semi-online scheduling · Offline scheduling · Makespan · Competitive ratio

MSC classification (2000) 90B35: Scheduling theory, deterministic

1 Introduction

Parallel machine scheduling has been studied extensively over the last 50 years. The problem, in general, can be described as follows. A set of n jobs $J = \{J_1, J_2, \ldots, J_n\}$ has to be scheduled on m parallel machines $M = \{M_1, M_2, \ldots, M_m\}$. Job J_i has a processing requirement p_i, and machine M_j operates at a speed $v_{i,j}$ when processing job J_i. The time it takes for job J_i to be processed by machine M_j is $\frac{p_i}{v_{i,j}}$. If $v_{i,j} = 1$ for all i and j, then the machines are referred to as *identical* machines. If $v_{i,j} = v_j$ for all i, then the machines are referred to as *uniform* machines. Finally, if $v_{i,j}$ is totally arbitrary, then the machines are referred to as *unrelated* machines. According to the 3-field notation introduced by Graham et al. (1979), P, Q and R are used to denote identical, uniform and unrelated machines, respectively. The goal is to schedule these n jobs on the m machines so as to optimize a given objective function.

Many objective functions have been proposed and studied in the literature. The common ones include the makespan, the total weighted completion time, the number of tardy jobs, the maximum weighted tardiness, and the total weighted tardiness. The makespan is defined as the time it takes to complete all jobs. This will be the main objective function considered in this paper. According to the 3-field notation by Graham et al. (1979), the problems are denoted as $P \mid\mid C_{\max}$, $Q \mid\mid C_{\max}$, and $R \mid\mid C_{\max}$.

In recent years, parallel machine scheduling has been studied under the so-called machine eligibility constraints. In such a model, job J_i cannot be processed on just any one of the m machines. Instead, it can only be processed on any machine that belongs to a specific subset of the m machines, namely subset $\mathcal{M}_i \subseteq M$, which is referred to as the *processing set* of job J_i. This problem will be called *parallel machine scheduling subject to machine eligibility constraints*. The three problems defined above with machine eligibility constraints will be denoted by $P \mid \mathcal{M}_i \mid C_{\max}$, $Q \mid \mathcal{M}_i \mid C_{\max}$, and $R \mid \mathcal{M}_i \mid C_{\max}$. Clearly, $R \mid \mathcal{M}_i \mid C_{\max}$ is a special case of $R \mid\mid C_{\max}$. It is also easy to see that $P \mid \mathcal{M}_i \mid C_{\max}$ and $Q \mid \mathcal{M}_i \mid C_{\max}$ are special cases of $R \mid\mid C_{\max}$, since we can set $v_{i,j}$ equal to zero if job J_i cannot be processed by machine M_j. In the remainder of this paper, we will only focus on $P \mid \mathcal{M}_i \mid C_{\max}$ and $Q \mid \mathcal{M}_i \mid C_{\max}$. Figure 1 depicts the complexity hierarchy of the machine environments.

In this paper, we discuss online scheduling problem subject to eligibility constraints. An algorithm for an online problem does not have access to the entire input instance, unlike algorithms for offline problems which have access to all input information in advance. Instead, it obtains the input piece by piece, and has to react to new requests

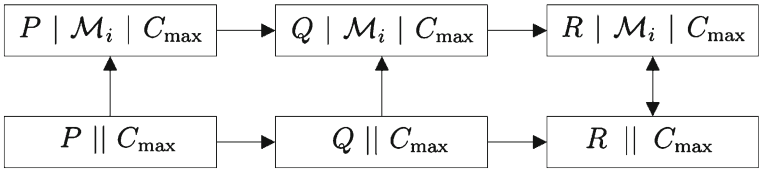

Fig. 1 The complexity hierarchy of the machine environments

with only a partial knowledge of the input (see Sgall 1998). We deal with two different online scheduling paradigms, namely online scheduling over list and online scheduling over time. Under the paradigm of online over list, jobs arrive one by one and each of them has to be scheduled immediately at its arrival without any information regarding subsequent jobs. Under the paradigm of online over time, jobs arrive over time and they can be started either immediately or after some delay without any knowledge regarding jobs that arrive in the future.

The organization of the paper is as follows. In the next section we will present the preliminaries regarding eligibility constraints and online scheduling. In Section 3 we review the results regarding online over list and in Section 4 we review the results regarding online over time. Finally, we point out some open problems for future research.

2 Preliminaries

2.1 Machine eligibility

The processing set of a job, a non-empty subset of all the machines, may be arbitrary or may have some structure. Depending upon the structure of the processing sets, there are four special classes of eligibility constraints that researchers have studied extensively:

(i) tree-hierarchical processing sets,
(ii) Grade of Service (GoS) processing sets,
(iii) interval processing sets, and
(iv) nested processing sets.

With tree-hierarchical processing sets, each machine is represented by a node, and the nodes are connected in the form of a rooted tree. Each job is associated with a machine node and the processing set of a job is the set of machines consisting of the node and all the nodes that are on the unique path from the associated node to the root of the tree. Tree-hierarchical processing sets for identical and uniform machines will be denoted by $P \mid \mathcal{M}_i(tree) \mid C_{\max}$ and $Q \mid \mathcal{M}_i(tree) \mid C_{\max}$, respectively. Figure 2 shows an example of tree-hierarchical processing sets. In this example, machine M_5 is the root of the tree. Machines M_1, M_2 and M_4 are the leaves of the tree. Jobs J_1 and J_2 are associated with the leaf machine M_1, and their processing set is $\{M_1, M_3, M_5\}$. Jobs J_3 and J_4 are associated with the leaf machine M_2, and their processing set is

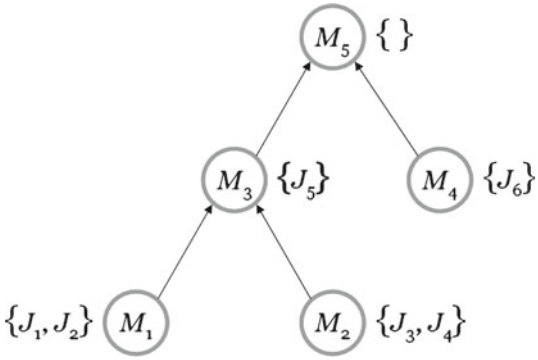

Fig. 2 Illustrating the tree-hierarchical processing set: $\mathcal{M}_1 = \mathcal{M}_2 = \{M_1, M_3, M_5\}$; $\mathcal{M}_3 = \mathcal{M}_4 = \{M_2, M_3, M_5\}$; $\mathcal{M}_5 = \{M_3, M_5\}$; $\mathcal{M}_6 = \{M_4, M_5\}$

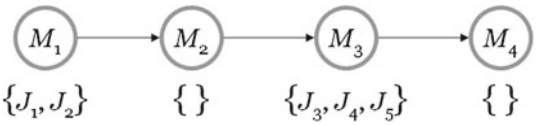

Fig. 3 Illustrating the GoS processing set: $\mathcal{M}_1 = \mathcal{M}_2 = \{M_1, M_2, M_3, M_4\}$; $\mathcal{M}_3 = \mathcal{M}_4 = \mathcal{M}_5 = \{M_3, M_4\}$

$\{M_2, M_3, M_5\}$. Job J_5 is associated with the machine M_3 and its processing set is $\{M_3, M_5\}$. Finally, J_6 is linked to the machine M_4, and its processing set is $\{M_4, M_5\}$.

A special case of the tree-hierarchical processing set structure is the so-called Grade of Service(GoS) processing set structure. With GoS processing sets, the form of the rooted tree is simply a chain. This eligibility is motivated by the customer differentiation scheme in the service industry in Hwang et al. (2004b). When a service provider has customers categorized as platinum, gold, silver, and regular members, with higher-level customers receiving better services, one possible way of providing such differentiated service is to label servers (i.e., machines) and customers (i.e., jobs) with prespecified grade of service (GoS) levels and allow a customer to be served by a server only when the GoS level of the customer is no less than the GoS level of the server. Thus, when all the machines are linearly ordered according to their grades, the eligible set of job J_i always can be expressed as $\{1, 2, \ldots, g_i\}$, where g_i is a job-dependent parameter. The problem under GoS eligibility constraints will be denoted by $P \mid \mathcal{M}_i(GoS) \mid C_{\max}$ and $Q \mid \mathcal{M}_i(GoS) \mid C_{\max}$ for identical and uniform machines, respectively. Figure 3 shows an example of GoS processing sets.

With interval processing sets, the machines are linearly ordered, say $M_1, \ldots, M_m$. Associated with each job J_i are two machine indexes, a_i and b_i. The processing set of job J_i consists of machines $M_{a_i}, M_{a_i+1}, \ldots, M_{b_i}$. Interval processing sets for identical and uniform machines will be denoted by $P \mid \mathcal{M}_i(interval) \mid C_{\max}$ and $Q \mid \mathcal{M}_i(interval) \mid C_{\max}$, respectively. Figure 4 shows an example of an interval processing set structure. In this example, the processing set of J_1 is $\{M_1, M_2, M_3\}$, and the processing set of J_4 is $\{M_4, M_5\}$. The processing set of J_2 and J_3 is $\{M_2, M_3, M_4\}$.

A special case of an interval processing set structure is a nested processing set structure. With nested processing sets, for any pair of jobs J_i and J_k, we either have $\mathcal{M}_i \subseteq \mathcal{M}_k$, or $\mathcal{M}_k \subseteq \mathcal{M}_i$, or $\mathcal{M}_i \cap \mathcal{M}_k = \emptyset$. Nested processing sets for identical and uniform machines will be denoted by $P \mid \mathcal{M}_i(nested) \mid C_{\max}$ and $Q \mid \mathcal{M}_i(nested) \mid C_{\max}$, respectively. Moreover, a special case of a nested processing set structure is a

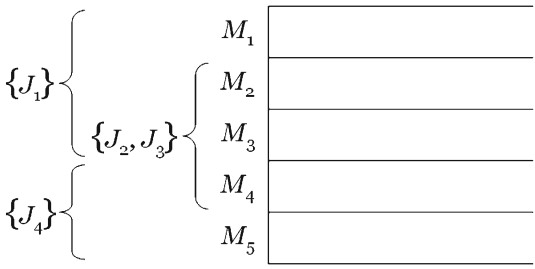

Fig. 4 Illustrating the interval processing set:
$\mathcal{M}_1 = \{M_1, M_2, M_3\}$;
$\mathcal{M}_2 = \mathcal{M}_3 = \{M_2, M_3, M_4\}$;
$\mathcal{M}_4 = \{M_4, M_5\}$

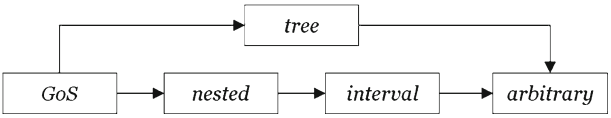

Fig. 5 The complexity hierarchies between different eligibilities

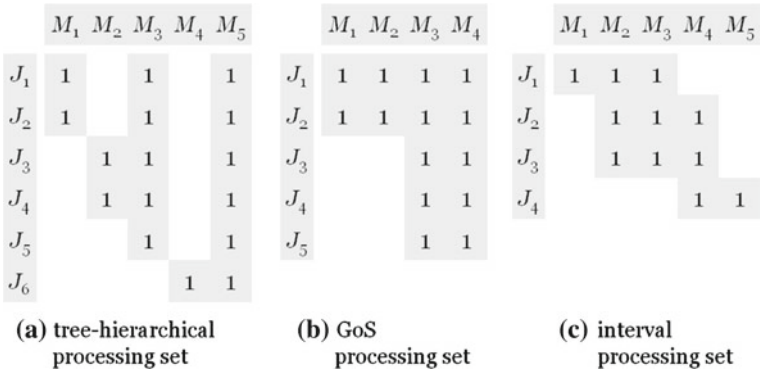

Fig. 6 Incident matrices of illustrating examples in Figs. 2, 3, and 4

GoS processing set structure. Figure 5 depicts the complexity hierarchies between the different sets of machine eligibility constraints.

We consider a job-machine incident matrix X which is a $(n \times m)$-matrix and whose element x_{ij} is 1 if job J_i is eligible to machine M_j, 0 otherwise. Thus, each row represents each job and each column represents each machine. (see Fig. 6).

It may be possible to reorder the columns (rows) such that the ones in every row (column) are consecutive. Interval eligibility has, by definition, an incident matrix with consecutive ones in the rows.

Interestingly, the tree-hierarchical eligibility has an incident matrix with consecutive one in the columns. To find a proper sequence, (i) we sequence all nodes according to postorder traversal sequence in depth-first search and (ii) we assign all jobs to their associated nodes in the sequence. For example, the postorder traversal sequence of nodes is $\langle M_1, M_2, M_3, M_4, M_5\rangle$ and thus, the sequence of jobs is $\langle J_1, J_2, J_3, J_4, J_5, J_6\rangle$.

Moreover, we define E_j to be the set of jobs that can be processed on machine M_j. This E_j satisfies the nested property. Suppose two sets E_j and E_h have a nonempty

intersection. Then, we may assume that machines M_j and M_h have a commonly eligible job J_i. This implies that machines M_j and M_h lie on the path from the machine associated with job J_i to the root of the tree and, thus, one of the two machines is an ancestor of the other or they are the same. Thus, E_j satisfies the nested property. Furthermore, the intersection of nested processing sets and tree-hierarchical processing sets is exactly GoS eligibility.

Scheduling problems with eligibility constraints occur quite often in practice. In manufacturing and logistics industry, Glass and Mills (2006) describe an application of nested processing in the drying stage of a flour mill in the United Kingdom and Ou et al. (2008) consider cranes with weight limit for loading and unloading cargoes of a vessel and find that crane restriction of handling each piece of cargo can be modelled into GoS eligibility constraints. In service provision application, Hwang et al. (2004b) present a differentiation scheme for different grade of service level customers and Wang and Xing (2010) analyze the worst-case performance for various on-line service polices. Data processing and transmission in computer science is also a good example for application of eligibility constraints. Azar et al. (1995) study an online load balancing with assignment restriction and Bar-Noy et al. (2001) focus on online load balancing among severs with a specific network topology. More applications can be found in the survey paper by Leung and Li (2008).

2.2 Offline, online and semi-online scheduling

In the literature researchers have studied the above scheduling problems in *offline* settings as well as in *online* settings. In offline scheduling, the scheduler has perfect information with regard to the job characteristics, such as arrival times, processing requirements, and processing sets of jobs. This information is given to the scheduler before he constructs a schedule. In contrast, in online scheduling, the characteristics of a job are not known to the scheduler before the job's arrival. However, at its arrival, all job characteristics become known to the scheduler. This version of online scheduling is at times also referred to as clairvoyant online scheduling. This is in contrast to non-clairvoyant online scheduling, where the processing requirement of a job only becomes known after its processing has been completed. In this paper we do not consider non-clairvoyant online scheduling. For a general overview of online scheduling, please see the survey papers of Sgall (1998) and Pruhs et al. (2004). For more information on online algorithms, the paper of Albers (2003) may be useful.

Two types of online scheduling problems are being considered:

(i) online over list, and
(ii) online over time.

In online over list, the jobs are ordered in some list (sequence) and are given to the scheduler one at a time according to this list. When a job is given to the scheduler, the scheduler knows all its characteristics including the processing requirement; he has to assign it immediately to some machine and some time slot, without being able to observe the remaining jobs on the list. Any assignment is irrevocable.

In online over time, jobs arrive at arbitrary points in time and the scheduler has no information regarding the jobs that have not arrived yet. Thus the online feature in this

case is the lack of information concerning the jobs that arrive in the future. When a job arrives, the scheduler knows all its characteristics including its processing requirement and he has the option of scheduling it immediately or postpone its scheduling till some later point in time.

Throughout the paper, the symbols $P \mid \mathcal{M}_i \mid C_{\max}$ and $Q \mid \mathcal{M}_i \mid C_{\max}$ will be used to denote online over list for identical and uniform machines, respectively. We use the symbols $P \mid r_i, \mathcal{M}_i \mid C_{\max}$ and $Q \mid r_i, \mathcal{M}_i \mid C_{\max}$ to denote online over time for identical and uniform machines, respectively.

In addition, researchers have also studied the so-called *semi-online* problems, where some partial information regarding the jobs is given to the scheduler before the scheduler constructs a schedule. Typical examples of partial information provided include the sum of the processing requirements of all jobs, the maximum processing requirement of the jobs, the optimal objective function value, and any combinations of the previous ones. In this paper we will focus on online and semi-online algorithms.

When applying preemptive options of offline problems to online problems, we need some modification. The preemptive option is an assumption that concerns whether a scheduler is allowed to interrupt the processing of a job at any time point and how to determine the remaining amount of processing requirement of the interrupted job. The following are three preemptive options for online problems. However, in online over list, only the first two options are applicable whereas in online over time, all three are applicable.

(i) non-preemption – preemption is not allowed.
(ii) preemption—a job can be interrupted and when an interrupted job is afterwards put back on the machine, it only needs the machine for its *remaining* processing requirement.
(iii) restart—a job can be interrupted and when an interrupted job is afterwards put back on the machine, it needs the machine for its *original* processing requirement.

According to the 3-field notation, the symbol for preemptive option appears in the second field. For non-preemptive case, no symbol appears while for the preemptive case and the restart case, $pmtn$ and $restart$ will appear, respectively.

Problems with jobs having identical processing requirements will also be considered. In online over list problems, jobs do not have release dates. So by scaling the processing requirements, we may assume that all processing requirements are unity. Thus, this restriction is denoted by $p_i = 1$ in the second field according to the 3-field notation. On the other hand, in online over time problems, jobs have release dates. In such a case, we assume that all release dates have integer values and all jobs have identical processing requirements. This restriction is denoted by $p_i = p$ in the second field of the 3-field notation. Obviously, a problem with $p_i = 1$ is a special case of a problem with $p_i = p$, and a problem with $p_i = p$ is a special case of a problem with arbitrary processing requirements.

For semi-online scheduling problems, the partial information also appears in the second field. When the total processing requirement, the maximum processing requirement, and the optimal objective function value are given, $semi(sum)$, $semi(max)$,

and $semi(opt)$ denote the cases, respectively. If more than one piece of information is given, then they appear in parentheses followed by $semi$; for example $semi(sum, max)$.

2.3 The scope of the paper

We focus on the theoretical analyses of online scheduling problems and their algorithms. All scheduling problems we consider ask for minimization of some objective function (performance measure).We use the competitive ratio, ρ, to evaluate the performance of an online algorithm. An online algorithm is ρ-competitive if for each input instance the objective value of the schedule produced by the algorithm is at most ρ times larger than the optimal objective value. Competitive ratio is the smallest value of ρ such that the algorithm is ρ-competitive. The competitive ratio may depend on m.

A lower bound for the competitive ratio of any algorithm for a specific problem is denoted by $\mathcal{L}(\rho)$, i.e., there cannot exist an algorithm better than ρ-competitive. If algorithm $\mathcal{A}$ for a problem has a competitive ratio that matches the lower bound for that problem, then $\mathcal{A}$ is referred to as an optimal algorithm.

There is a useful observation concerning a relationship between problems subject to eligibility constraints. If all problem instances of problem $\mathcal{P}_1$ are also problem instances of problem $\mathcal{P}_2$ and all feasible solutions of problem $\mathcal{P}_1$ are also feasible solutions of problem $\mathcal{P}_2$, then we say that problem $\mathcal{P}_1$ is a special case of problem $\mathcal{P}_2$. Then,

- the lower bound of the competitive ratio of problem $\mathcal{P}_1$ is less than or equal to the lower bound of the competitive ratio of problem $\mathcal{P}_2$ and
- the upper bound of the competitive ratio of problem $\mathcal{P}_1$ is less than or equal to the upper bound of the competitive ratio of problem $\mathcal{P}_2$.

Thus, we can make the following observations.

Observation 1 $Pm \mid \mathcal{M}_i \mid C_{\max}$ is a special case of $Pm+1 \mid \mathcal{M}_i \mid C_{\max}$. A problem instance of $Pm \mid \mathcal{M}_i \mid C_{\max}$ can serve as a problem instance of $Pm+1 \mid \mathcal{M}_i \mid C_{\max}$, with no jobs eligible to machine M_{m+1}.

Observation 2 If the eligibility of problem $\mathcal{P}_1$ is a special case of the eligibility of problem $\mathcal{P}_2$, problem $\mathcal{P}_1$ is a special case of problem $\mathcal{P}_2$.

According to our definition, $Pm \mid\mid C_{\max}$ is not a special case of $Pm+1 \mid\mid C_{\max}$.

For online over time, we consider another relationship between the problem with non-preemptive jobs and the problem with restart jobs. In both cases, the optimal offline makespan values are the same. Since a non-preemptive algorithm can function as an algorithm allowing restarts, we can make the following observation.

Observation 3 The upper bound of the competitive ratio of the problem with non-preemptive jobs is greater than or equal to the upper bound of the competitive ratio of the problem with restart jobs.

As stated earlier, we study two streams of online scheduling problems, namely online over list and online over time. In each stream, we deal with a variety of cases with different machine environments, different kinds of eligibilities, and different preemptive options.

3 Online over list

In this section we consider online scheduling over list. We assume that the machine environment consists of identical machines, unless stated otherwise. In the next three subsections, we consider arbitrary eligibility, tree-hierarchical eligibility, and GoS eligibility. In Sect. 3.4, we consider GoS eligibility with two levels. Finally, in Sect. 3.5, we consider interval eligibility.

3.1 Arbitrary eligibility

Arbitrary eligibility means that there is no special structure on the processing sets and each processing set is an arbitrary nonempty subset of the machine set. The first result in online scheduling subject to arbitrary machine eligibility constraints is due to Azar et al. (1995). They consider the following list scheduling algorithm known as $\mathcal{AW}$: when a job arrives the algorithm assigns it to an eligible machine that has the smallest current load among all eligible machines. They show that this list scheduling algorithm achieves a competitive ratio not greater than $\lceil \log_2 m \rceil + 1$. They also show that the competitive ratio of *any* online algorithm is at least $\lceil \log_2 (m+1) \rceil$. Thus, this algorithm is optimal when m is a power of two with a competitive ratio of $\log_2 m + 1$. Hwang et al. (2004a) give an improved analysis of the same algorithm and show that the competitive ratio is not greater than $\log_2 \frac{4m}{\lambda} - \frac{1}{\lambda}$, where λ is the number of machines eligible for processing the job with the latest completion time. When $\lambda = 1$, it becomes $\log_2 m + 1$, which is a slightly improved upper bound. Based on the above results, algorithm $\mathcal{AW}$ is known to be optimal 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. Lim et al. (2010) construct a new competitive ratio for algorithm $\mathcal{AW}$ and a new lower bound for the competitive ratio of the problem. For the problem with m machines, the proposed competitive ratio of algorithm $\mathcal{AW}$ is $\lfloor \log_2 m \rfloor + m/2^{\lfloor \log_2 m \rfloor}$ and the proposed lower bound for the competitive ratio, denoted by $LB(m)$, is presented by the following recursive formula:

$$LB(m) = \begin{cases} 1 & m = 1 \\ LB(\gamma_m) + \lceil \gamma_m/(m-\gamma_m) \rceil^{-1} & m \geq 2, \end{cases}$$

where

$$\gamma_m = \arg \max_{\lceil m/2 \rceil \leq i \leq m-1} \left\{ LB(i) + \lceil i/(m-i) \rceil^{-1} \right\}.$$

They show that the gap is no more than an irrational number which is approximately 0.1967. Furthermore, they 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 $k \neq k'$. They further analyze the case with seven machines showing that their gap is no more than 1/180 (≈ 0.00556).

Park et al. (2006) study online and semi-online scheduling on two machines. (Note that for two machines, arbitrary eligibility is equivalent to nested eligibility.) For

arbitrary eligibility, the competitive ratio of *any* online algorithm cannot be more than 2. This is because the makespan of any schedule cannot be more than the total processing requirement, which cannot be more than twice the optimal makespan. On the other hand, the competitive ratio of any online algorithm cannot be less than 2 due to the following example. Consider two jobs with unit processing requirement. The first job can be scheduled on both machines, while the second job is only eligible on the machine to which the first job is assigned. Therefore, any online algorithm would be an optimal algorithm with a competitive ratio of 2.

This lower bound example also applies to any semi-online problem with partial information of total processing requirement, the maximum processing requirement, and the optimal makespan. This idea can be further generalized to the semi-online problem with an arbitrary number of machines subject to arbitrary processing sets. Azar et al. (1995) provide a lower bound example for the case with $m = 2^k$ machines, where all processing requirements are unities, and in the optimal schedule each machine processes only one job and thus the optimal makespan is a unity. Based on this example, even if we know the total processing requirement, the maximum processing requirement, and even the optimal makespan, such partial information is not helpful to develop a better algorithm.

In case the number of distinct processing sets is restricted to 2 and the processing requirements are unity, i.e., $P \mid p_i = 1, |\{\mathcal{M}_i \mid J_i \in J\}| = 2 \mid C_{\max}$, then there is an optimal online algorithm with a competitive ratio of 1 by Mandelbaum and Shabtay (2010). However, for the case where there are more than three distinct processing sets a 1-competitive online algorithm cannot possibly exist.

For the uniform machine case, Lee et al. (2009) consider the problem $Q2 \mid \mathcal{M}_i \mid C_{\max}$. They assume that $v_1 = 1$ and $v_2 = s$. Since there are only two machines, by symmetry, we may assume that $s > 1$. They give an optimal online algorithm, the so-called High Speed Machine First (HSF) algorithm, with a competitive ratio of $1 + 1/s$.

3.2 Tree-hierarchical eligibility

Bar-Noy et al. (2001) consider online scheduling subject to tree-hierarchical eligibility. They present a deterministic (as opposed to a randomized) algorithm with a 4-competitive ratio for equal-processing-requirement jobs, and a 5-competitive algorithm for unequal-processing-requirement jobs. They also show that randomizing their algorithm improves its competitiveness to e and $e + 1$ for equal-processing-requirement and unequal-processing-requirement jobs, respectively, where e is the base of natural logarithm ($e \approx 2.7183$).

3.3 GoS eligibility

Bar-Noy et al. (2001) consider online scheduling subject to GoS eligibility. They give a deterministic algorithm with a competitive ratio of e and $e + 1$ for equal-processing-requirement and unequal-processing-requirement jobs, respectively. Moreover, they show a lower bound of e for the competitive ratio of *any* online algorithm

for equal-processing-requirement jobs. However, their upper and lower bounds for the optimal competitive ratios assume that the number of machines is infinite. Thus, there are several results with regard to cases that have a specific number of machines.

Park et al. (2006) and Jiang et al. (2006) independently consider GoS eligibility with two machines. In both papers, the authors first show that *any* online algorithm must have a competitive ratio of at least 5/3 and then present an online algorithm with a competitive ratio of 5/3. Thus, the algorithm is optimal.

Tan and Zhang (2010b) give a slight improvement of the algorithm of Bar-Noy et al. (2001). They observe that the algorithm of Bar-Noy et al. (2001) involves first the use of an LP-program to find a solution for the *fractional* model (i.e., the model where a job can be split and assigned to several machines simultaneously), and then a *de-fractionalization* of the LP program to obtain a solution for the original problem. Tan and Zhang (2010b) give a more effective LP algorithm for the fractional model, with a better competitive ratio than the algorithm of Bar-Noy et al. (2001) for all values of m. For $m = 4$ and 5, they present improved algorithms with competitive ratios of 2.333 and 2.610, respectively.

Park et al. (2006) consider a semi-online problem on two machines when the total processing requirement is given. They first show that *any* online algorithm must have a competitive ratio of at least $3/2$ and then provide a semi-online algorithm with a competitive ratio of $3/2$, implying the algorithm is in fact optimal.

Wu and Yang (2010) consider two semi-online problems where in the first problem the optimal makespan is known and in the second problem the largest processing requirement is known. For the first problem, they first show that *any* online algorithm must have a competitive ratio of at least $3/2$, and provide an optimal algorithm with a competitive ratio of 3/2. For the second problem, they first show that *any* online algorithm must have a competitive ratio of at least $(\sqrt{5}+1)/2$, and provide an optimal algorithm with a competitive ratio of $(\sqrt{5}+1)/2$.

Liu et al. (2010) study semi-online scheduling with two machines and the jobs subject to GoS machine eligibility. Their definition of semi-online scheduling is somewhat different from the earlier definition. Specifically, they study two problems. The first problem is concerned with bounded processing requirement constraints. That is, the processing requirement p_i of job J_i is bounded by an interval $[a, \alpha a]$ with $\alpha > 1$. The second problem assumes that, in addition to the bounded processing requirement constraints, the total processing requirement of all jobs is known in advance.

For the first problem, Liu et al. (2010) obtain the following lower bound for the competitive ratio:

$$\mathcal{L}(\rho) = \begin{cases} \frac{1+\alpha}{2} & \text{for } 1 < \alpha < 2, \\ \frac{3}{2} & \text{for } 2 \le \alpha < 5, \\ \frac{4+\alpha}{6} & \text{for } 5 \le \alpha < 6. \end{cases}$$

(Note that for the case $\alpha \ge 6$, Park et al. (2006) had already provided a lower bound of $\frac{5}{3}$). They, furthermore, propose an algorithm, the so-called B-ONLINE algorithm,

with the competitive ratio:

$$\rho(\text{B-ONLINE}) = \begin{cases} \frac{1+\alpha}{2} & \text{for } \frac{25}{14} < \alpha < 2, \\ \frac{3}{2} & \text{for } 2 \leq \alpha < 5, \\ \frac{4+\alpha}{6} & \text{for } 5 \leq \alpha < 6 \text{ only when the optimal makespan } \geq 20a. \end{cases}$$

Thus, B-ONLINE is an optimal algorithm in the given range of α. Note that B-ONLINE is actually a modification of the ONLINE algorithm proposed by Park et al. (2006), which has a competitive ratio of 5/3. B-ONLINE has the same competitive ratio of 5/3 as ONLINE for $\alpha \geq 6$.

For the second problem, Liu et al. (2010) show a lower bound of $\frac{1+\alpha}{2}$ for $1 < \alpha < 2$. (Note that for $\alpha \geq 2$, Park et al. (2006) had already obtained a lower bound of $\frac{3}{2}$) They then propose an algorithm, the so-called B-SUM-ONLINE algorithm, and show that it has a competitive ratio of $\frac{1+\alpha}{2}$ when $1 < \alpha < 2$ and the total processing requirement of all jobs is at least $\frac{2\alpha a}{\alpha-1}$. (Note that B-SUM-ONLINE is actually a modification of the SEMI-ONLINE algorithm proposed by Park et al. (2006). Both algorithms have the same competitive ratio of 3/2 for $\alpha \geq 2$).

Jiang et al. (2006) consider GoS eligibility with two machines when preemption is allowed but no idle time between consecutive preempted parts of a job is allowed. They present an online algorithm with a competitive ratio of 3/2. Furthermore, they show that 3/2 is a lower bound. Thus, the algorithm is in fact optimal.

Dosa and Epstein (2008) consider problem $P3 \mid \mathcal{M}_i(GoS), pmtn \mid C_{\max}$. They present a $\frac{3}{2}$-competitive algorithm and show that no online algorithm can have a competitive ratio better than $\frac{3}{2}$. Thus, their algorithm is optimal for three identical machines.

Liu et al. (2009) consider problem $Q2 \mid \mathcal{M}_i(GoS) \mid C_{\max}$ with the two machines having different speeds. They provide lower and upper bounds with an algorithm for the competitive ratio as a function of the ratio of the two speeds. Lee et al. (2009) point out an error in Liu et al. (2009) and improve both lower and upper bounds. Finally, Tan and Zhang (2010a) provide an optimal algorithm. They assume that $v_1 = s$ and $v_2 = 1$. Note that s can be either smaller or larger than 1. For $0 < s < 1$, they give an optimal algorithm with a competitive ratio of $\min\{1 + s, 1 + \frac{1+s}{1+s+s^2}\}$. For $s > 1$, they give an optimal algorithm with a competitive ratio of $\min\{1 + \frac{1}{s}, 1 + \frac{2s}{1+s+s^2}\}$.

Dosa and Epstein (2008) consider problem $Q2 \mid \mathcal{M}_i(GoS), pmtn \mid C_{\max}$. They consider the two cases: (1) $v_1 \geq v_2$ and (2) $v_2 \geq v_1$. In the first case, they provide a $\frac{s(s+1)^2}{s^3+s^2+1}$-competitive algorithm that is optimal, where $s = \frac{v_1}{v_2}$. In the second case, they provide a $\frac{(s+1)^2}{s^2+s+1}$-competitive algorithm that is optimal, where $s = \frac{v_2}{v_1}$.

3.4 GoS eligibility with two GoS levels

GoS eligibility with two GoS levels, i.e., a special case of GoS eligibility, implies that there are two groups of jobs and two groups of machines. Assume that the first k machines have GoS level 1 and the remaining $m - k$ machines have GoS level 2.

Each job has GoS level of 1 or 2. Thus, machines with GoS level 1 can process any job whereas machines with GoS level 2 can only process jobs with GoS level 2. Let this problem be denoted by $P \mid \mathcal{M}_i(2GoS(k)) \mid C_{\max}$.

Jiang (2008) studies online scheduling on parallel machines with two GoS levels. He first shows that any online algorithm must have a competitive ratio of at least 2. He then provides an online algorithm with a competitive ratio of $(12 + 4\sqrt{2})/7 \approx 2.522$.

Zhang et al. (2009) also study online scheduling on m machines with two GoS levels. They propose an improved online algorithm TSL with a competitive ratio of

$$1 + \frac{m^2 - m}{m^2 - km + k^2} < \frac{7}{3}.$$

They also present another algorithm SLS with a competitive ratio of

$$1 + \min\left\{\frac{k-1}{\lfloor l_0 \rfloor}, \frac{m-1}{m - \lceil l_0 \rceil}\right\} \quad \text{where} \quad l_0 = \frac{m(k-1)}{m+k-2}.$$

Algorithm SLS outperforms algorithm TSL for some pairs of k and m. They study lower bounds for different pairs of k and m.

When the processing requirement is restricted to one, we can apply the result by Mandelbaum and Shabtay (2010). They prove that the problem with only two different processing sets has a 1-competitive algorithm. Thus, $P \mid \mathcal{M}_i(2GoS(k)), p_i = 1 \mid C_{\max}$ also has a 1-competitive algorithm.

3.5 Interval eligibility

Bar-Noy et al. (2001) consider online scheduling with interval eligibility. They provide a lower bound of $\Omega(\log m)$ for *any* online algorithm. More precisely, they provide a problem instance with a completion time that is at least $\frac{1}{2}\log_2 m$ times the optimal makespan when the number of machines is a power of two. The algorithm of Azar et al. (1995) implies that the competitive ratio of an optimal algorithm for interval eligibility lies in $\left[\frac{1}{2}\log_2 m,\ 1 + \log_2 m\right]$.

4 Online over time

This section deals with online parallel machine scheduling with jobs arriving over time and all the information regarding a job being revealed only upon its arrival. For job J_i, its release date is denoted by r_i. In the schedule, S_i and C_i denote the starting time and completion time of job J_i, respectively. Let $C_{\max}(\pi)$ and $C_{\max}(\sigma)$ denote the optimal makespan and the makespan of the schedule in the context, respectively.

We discuss in Sect. 4.1 the algorithms and lower bounds for problems in which the makespan has to be minimized. Online service scheduling, which is a variant of online scheduling over time, is discussed in Sect. 4.2. In Sect. 4.3, we consider results with regard to the total weighted completion time objective and related problems.

4.1 Makespan

First we consider results with regard to online scheduling problems over time with the makespan objective and no eligibility constraints. For the identical machine environment, $P \mid r_i \mid C_{\max}$, with preemption not being allowed, Chen and Vestjens (1997) prove that LPT has a 1.5-competitive ratio. They also provide lower bounds of 1.3473 and 1.3820 for the problems with m and two machines, respectively. Noga and Seiden (2001) prove that 1.3820 is the best competitive ratio for problem $P2 \mid r_i \mid C_{\max}$ by giving an optimum algorithm. When preemption is allowed, Hong and Leung (1992) provide a polynomial time algorithm for problem $P \mid r_i, pmtn \mid C_{\max}$ with a competitive ratio of 1. Thus, their algorithm is optimal.

Surprisingly, there are very few results concerning online scheduling subject to eligibility constraints with the makespan as objective when jobs arrive over time. Lee et al. (2010a) solve two problems concerning online scheduling of equal-length jobs on two machines subject to arbitrary eligibility constraints and GoS eligibility constraints, i.e., $P2 \mid r_i, \mathcal{M}_i, p_i = p \mid C_{\max}$ and $P2 \mid r_i, \mathcal{M}_i(GoS), p_i = p \mid C_{\max}$. They devise optimal algorithms for both problems and the competitive ratios are $(1+\sqrt{5})/2$ and $\sqrt{2}$, respectively.

There is an important study of online scheduling problems in connection with offline scheduling problems by Shmoys et al. (1995). They state and prove the following theorem, which provides general upper bounds of our problems.

Theorem 1 *Let $\mathcal{A}$ be a polynomial-time algorithm that is applicable in an environment where each job is available at time 0 and that always generates a schedule of a length of at most $\rho C_{\max}(\pi)$. For the analogous environment in which the existence of a job only becomes known upon its release date, there exists another polynomial-time algorithm $\mathcal{A}'$ that is applicable in this more general setting and that generates a schedule of a length of at most $2\rho C_{\max}(\pi)$.*

Proof By Shmoys et al. (1995). Let $\mathcal{I}$ be an instance that includes jobs with unknown release dates and let J^0 be the set of jobs available at time 0. The scheduler applies algorithm $\mathcal{A}$ and schedules the jobs in J^0, finishing at time f_0. Let J^1 be the set of jobs released in the time interval $(0, f_0]$. The scheduler now, at time f_0, applies algorithm $\mathcal{A}$ to schedule J^1, finishing at time f_1. In general, let J^{i+1} be the set of jobs released in the time interval $(f_{i-1}, f_i]$, and let f_i be the point in time when the schedule for J^i completes. At time f_i, the scheduler uses algorithm $\mathcal{A}$ to schedule the jobs in J^{i+1}. Let f_k be the finishing time of the entire schedule.

To analyze the length of the resulting schedule, consider the modified problem instance $\mathcal{I}'$ in which jobs in J^k are released at time f_{k-2}. Since these jobs are released in $\mathcal{I}'$ at an earlier point in time than in $\mathcal{I}$, it follows that $C_{\max}(\pi(\mathcal{I}')) \leq C_{\max}(\pi(\mathcal{I}))$, where $\pi(\mathcal{I}')$ and $\pi(\mathcal{I})$ are optimal schedules for instances $\mathcal{I}'$ and $\mathcal{I}$, respectively.

Now note that $f_{k-2} + f_k - f_{k-1} \leq \rho C_{\max}(\pi(\mathcal{I}'))$, since the jobs in J^k are not released until f_{k-2} and the properties of algorithm $\mathcal{A}$ guarantee that $f_k - f_{k-1}$ is within a factor ρ of the shortest schedule for J^k. Similarly, $f_{k-1} - f_{k-2} \leq \rho C_{\max}(\pi(\mathcal{I}'))$. Therefore, $f_k \leq 2\rho C_{\max}(\pi(\mathcal{I}')) \leq 2\rho C_{\max}(\pi(\mathcal{I}))$. □

This theorem says that either an optimal or an approximation algorithm for offline problems can be used to develop an online algorithm for the problem where jobs

arrive over time. Thus, we need to find optimal and approximation algorithms for offline problems. For the offline problem $P \mid \mathcal{M}_i \mid C_{\max}$, Lenstra et al. (1990) provide a 2-approximation algorithm utilizing a linear programming and a rounding technique. Shchepin and Vakhania (2005) improve the worst case performance bound to $2 - \frac{1}{m}$ using an optimal rounding, where m is the number of machines. For offline problems $P \mid \mathcal{M}_i, r_i, p_i = p \mid C_{\max}$ and $Q \mid \mathcal{M}_i, r_i, p_i = p \mid C_{\max}$, Lee et al. (2010a) have provided optimal algorithms that run in polynomial time. For offline problem $P \mid \mathcal{M}_i, pmtn \mid C_{\max}$, Lawler and Labetoulle (1978) prove that it can be solved optimally in polynomial time.

Furthermore, offline problems $P \mid \mathcal{M}_i(GoS) \mid C_{\max}$ and $P \mid \mathcal{M}_i(nested) \mid C_{\max}$ have PTASs that are proposed by Ou et al. (2008) and Muratore et al. (2010), respectively. While the PTASs have very high running times, there are faster approximation algorithms with constant worst-case bounds for $P \mid \mathcal{M}_i(GoS) \mid C_{\max}$ proposed by Ou et al. (2008) and for $P \mid \mathcal{M}_i(nested) \mid C_{\max}$ proposed by Huo and Leung (2010a), Huo and Leung (2010b). The offline problem $P \mid \mathcal{M}_i(tree) \mid C_{\max}$ has a 4/3-approximation algorithm that runs in polynomial time by Huo and Leung (2010b). Therefore, based on Theorem 1, we can obtain several corollaries concerning the upper bounds of the competitive ratio.

Corollary 1

(i) *The online scheduling problem* $P \mid r_i, \mathcal{M}_i \mid C_{\max}$ *has a* $(4 - \frac{2}{m})$*-competitive polynomial time algorithm.*
(ii) *The online scheduling problem* $P \mid r_i, \mathcal{M}_i, p_i = p \mid C_{\max}$ *has a 2-competitive polynomial time algorithm.*
(iii) *The online scheduling problem* $Q \mid r_i, \mathcal{M}_i, p_i = p \mid C_{\max}$ *has a 2-competitive polynomial time algorithm.*
(iv) *The online scheduling problem* $P \mid r_i, \mathcal{M}_i, pmtn \mid C_{\max}$ *has a 2-competitive polynomial time algorithm.*
(v) *The online scheduling problem* $P \mid r_i, \mathcal{M}_i(GoS) \mid C_{\max}$ *has a* $(2 + \epsilon)$*-competitive polynomial time algorithm for any constant* ϵ*.*
(vi) *The online scheduling problem* $P \mid r_i, \mathcal{M}_i(nested) \mid C_{\max}$ *has a* $(2 + \epsilon)$*-competitive polynomial time algorithm for any constant* ϵ*.*
(vii) *The online scheduling problem* $P \mid r_i, \mathcal{M}_i(tree) \mid C_{\max}$ *has a* $(8/3)$ *-competitive polynomial time algorithm.*

Similar to Theorem 1, we have a theorem and a subsequent corollary for the restart case.

Theorem 2 *Let* $\mathcal{A}$ *be a polynomial-time algorithm that is applicable in an environment where each job is available at time 0 and that always generates a schedule of length at most* $\rho C_{\max}(\pi)$*. For the analogous environment in which the existence of a job only becomes known upon its release date, there exists another polynomial-time algorithm* $\mathcal{A}'$ *that is applicable in this more general setting and that generates a schedule of length at most* $(1 + \rho)C_{\max}(\pi)$*.*

Proof Let $\mathcal{I}$ be an instance including jobs with unknown release dates and let J^0 be the set of jobs available at time 0. Then the scheduler applies algorithm $\mathcal{A}$ and schedules

the jobs in J^0. Before completing all jobs in J^0, if the first arrival of a job or group of jobs occurs at time f_0, we restart running all jobs at time f_0. Let J^1 be the set of jobs released or restarted at time f_0. The scheduler now, at time f_0, applies algorithm $\mathcal{A}$ to the set of jobs J^1. In general let J^{i+1} be the set of jobs released or restarted at time f_i. At time f_i, the scheduler uses algorithm $\mathcal{A}$ to schedule the jobs in J^{i+1}. Let the last arrival time of jobs be t_{k-1} and let F be the finishing time of the entire schedule.

To analyze the length of the resulting schedule, consider the modified problem instance $\mathcal{I}'$ where jobs in J^k are released or restarted at time f_{k-1}. Since $J^k \subset \cup_{i=0}^{i=k} J^i$, we have $C_{\max}(\pi(\mathcal{I}')) \leq C_{\max}(\pi(\mathcal{I}))$, where $\pi(\mathcal{I}')$ and $\pi(\mathcal{I})$ are optimal schedules for instances $\mathcal{I}'$ and $\mathcal{I}$, respectively. Also, since there is at least one job that is released at time f_{k-1} in $\mathcal{I}'$, we have $f_{k-1} \leq C_{\max}(\pi(\mathcal{I}))$.

Now consider another problem instance $\mathcal{I}''$ that contains only jobs in J^k which are released at time zero and has an optimal makespan of $C_{\max}(\pi(\mathcal{I}''))$. Since algorithm $\mathcal{A}$ guarantees the worst case performance ratio of ρ,

$$F - f_{k-1} \leq \rho C_{\max}(\pi(\mathcal{I}'')) \leq \rho C_{\max}(\pi(\mathcal{I}')) \leq \rho C_{\max}(\pi(\mathcal{I})).$$

Therefore, $F \leq (1+\rho) C_{\max}(\pi(\mathcal{I}))$. □

Corollary 2 (i) *The online scheduling problem* $P \mid r_i, \mathcal{M}_i, restart \mid C_{\max}$ *has a* $(3 - \frac{1}{m})$*-competitive polynomial time algorithm.*

(ii) *The online scheduling problem* $P \mid r_i, \mathcal{M}_i, p_i = p, restart \mid C_{\max}$ *has a 2-competitive polynomial time algorithm.*

(iii) *The online scheduling problem* $Q \mid r_i, \mathcal{M}_i, p_i = p, restart \mid C_{\max}$ *has a 2-competitive polynomial time algorithm.*

(iv) *The online scheduling problem* $P \mid r_i, \mathcal{M}_i(GoS), , restart \mid C_{\max}$ *has a* $(2+\epsilon)$*-competitive polynomial time algorithm for any constant* ϵ*.*

(v) *The online scheduling problem* $P \mid r_i, \mathcal{M}_i(nested), restart \mid C_{\max}$ *has a* $(2+\epsilon)$*-competitive polynomial time algorithm for any constant* ϵ*.*

(vi) *The online scheduling problem* $P \mid r_i, \mathcal{M}_i(tree), restart \mid C_{\max}$ *has a* $(7/3)$*-competitive polynomial time algorithm.*

Before investigating the problems subject to eligibility constraints, we provide a lower bound for the problem without eligibility constraints when restarts are allowed. Shmoys et al. (1995) show that 10/9 = 1.111…is a lower bound for the competitive ratio of $P \mid r_i, restart \mid C_{\max}$. We present here a higher lower bound.

Lemma 1 *Any online algorithm for* $P \mid r_i, restart \mid C_{\max}$ *has a competitive ratio at least* $\sqrt{6}/2 \approx 1.2247$.

Proof We show this result by a set of adversary arguments. Recall that m is the number of machines and we may assume that $m \geq 2$.

Let $m = 2k$ $(2k+1)$ for m even (odd) and k be a positive integer. At time $t = 0$, $2k$ $(2k+1)$ jobs with processing requirement 1 arrive. These jobs are jobs of type 1. Since this model admits restarts, we do not have to consider any delay. Thus, all type 1 jobs start at time $t = 0$. At time $t = 3 - \sqrt{6}$, k jobs arrive with processing requirement $\sqrt{6} - 1$. These jobs are jobs of type 2.

We first consider the case where at least one type 2 job starts at time 1. It implies $C_{\max}(\sigma) \geq 1+(\sqrt{6}-1) = \sqrt{6}$. In the current optimal schedule, k type 1 jobs processed on the first k machines are interrupted and restarted on the other k machines at time 1 while all type 2 jobs start at their arrival time on the machines where the interrupted jobs were processed, implying $C_{\max}(\pi) = 2$. Thus, $C_{\max}(\sigma)/C_{\max}(\pi) \geq \sqrt{6}/2$.

Otherwise, all k type 2 jobs start at their arrival time implying that k type 1 jobs have to restart. Then k new jobs with processing requirement $\sqrt{6}-1$ arrive at time 1. Then, we have $C_{\max}(\sigma) \geq (3-\sqrt{6}) + (\sqrt{6}-1) + 1 = 3$. In an optimal schedule, all type 1 jobs start at time 0 and all jobs with processing requirement $\sqrt{6}-1$ start at time 1, implying $C_{\max}(\pi) = \sqrt{6}$. Thus, $C_{\max}(\sigma)/C_{\max}(\pi) = 3/\sqrt{6} \geq \sqrt{6}/2$. □

Note that this example works when $m = 2$. Thus, any online algorithm for $P2 \mid r_i, restart \mid C_{\max}$ has a competitive ratio of at least $\sqrt{6}/2$.

Lemma 2 *Any online algorithm for* $Pm \mid r_i, \mathcal{M}_i, pmtn \mid C_{\max}$ *has a competitive ratio at least* $1 + \frac{(m-1)(m-2)}{2m(2m-3)}$ *for* $m \geq 3$.

Proof We show this result by describing a class of adversary examples.

If a job is eligible only to machines M_k and M_m for $k \leq m-1$, then we call this job a job of type k. If a job is eligible to all machines, then we call this job a job of type m. At time zero, two jobs with processing requirement 1 arrive for each type k, $1 \leq k \leq m-1$; one type m job arrives with processing requirement 2. Note that in the current optimum schedule jobs of type k are scheduled on machine M_k, $k \leq m-1$, and the job of type m is scheduled on machine M_m, implying that the optimal makespan is exactly 2.

Let q_k be the total amount of processing of the jobs of type k until time $t = 1$ for all $1 \leq k \leq m$.

If there exists a type k job such that $q_k \leq \frac{2m^2-2m-2}{2m^2-3m}$ for $1 \leq k \leq m-1$, then the remaining amount of processing requirement of the type k jobs at time $t = 1$ is greater than or equal to

$$2 - \frac{2m^2-2m-2}{2m^2-3m} = \frac{2(m-1)^2}{2m^2-3m}.$$

Two jobs arrive at time $t = 1$ such that they have processing requirement of $m/(m-2)$ and they are eligible to machines M_k and M_m. We refer to them as type $m+1$ jobs. Since the remaining amount of type k and type $m+1$ jobs must be processed on machine M_k or M_m, the completion time on machine M_k or M_m is at least

$$1 + \frac{1}{2}\frac{2(m-1)^2}{2m^2-3m} + \frac{m}{m-2} = \left(1 + \frac{m}{m-2}\right) + \frac{(m-1)^2}{m(2m-3)}.$$

Furthermore, in the current optimal schedule,

1. Two jobs of type k are processed on machines M_k and M_m from time 0 to time 1,
2. Two jobs of type $m+1$ are processed on machine M_k and M_m from time 1 to time $1 + m/(m-2)$,

3. All jobs of type τ, for $\tau \neq k$ and $\tau \neq m$ are scheduled on machine M_τ, and
4. The job of type m is processed on all machines except machines M_k and M_m with a processing amount of $2/(m-2)$ in each machine.

Thus, the current optimal makespan is

$$C_{\max}(\pi) = 1 + \frac{m}{m-2}.$$

Thus,

$$\frac{C_{\max}(\sigma)}{C_{\max}(\pi)} \geq 1 + \frac{\frac{(m-1)^2}{m(2m-3)}}{\frac{2(m-1)}{m-2}} = 1 + \frac{(m-1)(m-2)}{2m(2m-3)}.$$

Otherwise, $q_k > \frac{2m^2-2m-2}{2m^2-3m}$ for all $1 \leq k \leq m-1$. Since $\sum_{k=1}^{m} q_k \leq m$, the remaining processing requirement of type m job is

$$2 - q_m > 2 - \left(m - (m-1)\frac{2m^2-2m-2}{2m^2-3m}\right) = 1 + \frac{(m-1)(m-2)}{m(2m-3)}.$$

If no jobs arrive later, then $C_{\max}(\sigma) \geq 1 + 1 + \frac{(m-1)(m-2)}{m(2m-3)}$ and since $C_{\max}(\pi) = 2$,

$$\frac{C_{\max}(\sigma)}{C_{\max}(\pi)} \geq 1 + \frac{(m-1)(m-2)}{2m(2m-3)}.$$

□

Corollary 3 *Any online algorithm for $P \mid r_i, \mathcal{M}_i, pmtn \mid C_{\max}$ has a competitive ratio at least* 5/4.

Proof Since $\lim_{m\to\infty} 1 + \frac{(m-1)(m-2)}{2m(2m-3)} = 5/4$, as m goes to infinity, the lower bound approaches 5/4. □

When the eligibility is restricted to the nested one, we have a different lower bound.

Lemma 3 *Any online algorithm for $Pm \mid r_i, \mathcal{M}_i(nested), pmtn \mid C_{\max}$ has a competitive ratio at least $1 + 1/8 = 1.125$ for $m = 2$ and $1 + 4/27 \approx 1.148$ for $m \geq 3$.*

Proof At time $t = 0$, $m+1$ jobs $J_1, \ldots, J_{m+1}$ arrive with $p_i = 1$ for $i = 1, \ldots, m$ and $p_{m+1} = m/(m-1)$, $\mathcal{M}_i = \{M_i\}$ for $i = 1, \ldots, m$ and $\mathcal{M}_{m+1} = \{M_1, \ldots, M_m\}$. Let q_i be the remaining processing requirement of job J_i at time 1. Obviously, $\sum_{i=1}^{m+1} q_i \geq m/(m-1)$.

If $q_i \geq \frac{m^2-m+1}{m^3}$ for some i, $i \in \{1, \ldots, m\}$, then job J_{m+2} arrives with $p_{m+2} = m/(m-1)^2$ and $\mathcal{M}_{m+2} = \{M_i\}$. The makespan of the schedule is at least

$1 + q_i + m/(m-1)^2$ while the current optimal makespan is $1 + m/(m-1)^2$. Thus, the competitive ratio is at least

$$1 + \frac{q_i}{1 + m/(m-1)^2} \geq 1 + \frac{\frac{m^2-m+1}{m^3}}{\frac{m^2-m+1}{(m-1)^2}} = 1 + \frac{(m-1)^2}{m^3}.$$

Otherwise, $q_i < \frac{m^2-m+1}{m^3}$ for all i, $i = 1, \ldots, m$ implying $q_{m+1} \geq m/(m-1) - \frac{m^2-m+1}{m^2}$. The makespan of the schedule is at least $1 + q_{m+1}$ while the current optimal makespan is $1 + 1/(m-1)$. Thus, the competitive ratio is at least

$$\frac{1 + q_{m+1}}{1 + 1/(m-1)} \geq \frac{1 + \frac{1}{m-1} + \frac{m-1}{m^2}}{1 + \frac{1}{m-1}} = 1 + \frac{(m-1)^2}{m^3}.$$

For $m = 2$, the proposed lower bound is 1+1/8. For $m \geq 3$, $1 + \frac{(m-1)^2}{m^3}$ has the maximum value of 1+4/27 at $m = 3$. By Observation 1, for $m \geq 3$, $Pm \mid r_i, \mathcal{M}_i(nested), pmtn \mid C_{\max}$ has a lower bound of 1+4/27. □

Note that this lower bound is better than the lower bound in Lemma 2 for arbitrary eligibility at $m = 3$. Thus, by Observation 2, $P3 \mid r_i, \mathcal{M}_i, pmtn \mid C_{\max}$ also has a lower bound of 1+4/27.

Lemma 4 *Any online algorithm for* $P \mid r_i, \mathcal{M}_i(GoS), pmtn \mid C_{\max}$ *has a competitive ratio at least* $1 + \phi^5 \approx 1.0917$, *where* $\phi = \frac{\sqrt{5}-1}{2}$.

Proof Assume that there are positive integers r and k satisfying $1 < r < k < m$ and $k \leq 2r$. At time zero, there are $2r$ jobs with processing requirement 1. They are eligible to machines $M_1, M_2, \ldots, M_k$. We refer to these jobs as jobs of type 1. At time zero, there are also $m - r$ jobs with processing requirement 2 that are eligible to all machines. We refer to these jobs as jobs of type 2. Note that in the current optimal schedule all the jobs of type 1 are scheduled on the first r machines and all the jobs of type 2 are scheduled on the remaining machines, implying that the current optimum makespan, $C_{\max}(\pi)$, is 2. Let Q be the amount of processing of type 2 jobs in machines $M_1, M_2, \ldots, M_k$ at time $t = \frac{2r}{k} \geq 1$.

If $Q \leq \frac{2r(k-r)}{m}$, then the average remaining processing requirement of type 2 jobs must be at least $2 - \frac{\frac{2r}{k}(m-k)+Q}{m-r}$. It implies that there exists a type 2 job with a remaining processing requirement at least $2 - \frac{\frac{2r}{k}(m-k)+Q}{m-r}$ at time $t = \frac{2r}{k}$. So, if no other job arrives, $C_{\max}(\sigma) \geq \frac{2r}{k} + 2 - \frac{\frac{2r}{k}(m-k)+Q}{m-r}$. Thus,

$$\frac{C_{\max}(\sigma)}{C_{\max}(\pi)} \geq \frac{\frac{2r}{k} + 2 - \frac{\frac{2r}{k}(m-k)+Q}{m-r}}{2} \geq 1 + \frac{r(k-r)(m-k)}{km(m-r)}.$$

Otherwise, k jobs arrive at time $\frac{2r}{k}$ with a processing requirement of $\frac{2(m-r)}{m-k} - \frac{2r}{k}$. They are eligible to machines $M_1, M_2, \ldots, M_k$ and we refer to them as type 3 jobs.

In an optimum schedule, all type 2 jobs must be processed on machines M_{k+1} to M_m and type 1 and type 3 jobs must be processed on machines M_1 to M_k, implying that the optimal makespan is $C_{\max}(\pi) = \frac{2(m-r)}{m-k}$. Since $Q > \frac{2r(k-r)}{m}$, the total remaining processing requirement of type 1 jobs at time $\frac{2r}{k}$, to be processed on machines M_1 to M_k, is at least $\frac{2r(k-r)}{m}$. All type 3 jobs must also be processed on machines M_1 to M_k. Thus,

$$C_{\max}(\sigma) \geq \frac{2r}{k} + \frac{2r(k-r)}{km} + \left(\frac{2(m-r)}{m-k} - \frac{2r}{k}\right) = \frac{2r(k-r)}{km} + \frac{2(m-r)}{m-k}.$$

Therefore,

$$\frac{C_{\max}(\sigma)}{C_{\max}(\pi)} \geq 1 + \frac{\frac{2r(k-r)}{km}}{\frac{2(m-r)}{m-k}} = 1 + \frac{r(k-r)(m-k)}{km(m-r)}.$$

In order to maximize the lower bound of the competitive ratio, we set $k = \phi m$ and $r = (1-\phi)m$ for large m, where $\phi = \frac{\sqrt{5}-1}{2}$. The competitive ratio is then $1 + \phi^5 \approx 1.0917$. □

The table below contains lower bounds for different values of m. Even for $m = 3$, there exists a lower bound example with a bound of $13/12 \approx 1.0833$.

m	r	k	Lower bound
3	1	2	1.08333
5	2	3	1.08889
8	3	5	1.09000
13	5	8	1.09014
21	8	13	1.09017

We can consider an interesting problem that is related to this problem. We want to find three positive integers a, b and c that maximize the value of $\frac{abc}{(a+b)(b+c)(a+b+c)}$. By setting $r = a$, $k = a + b$, and $m = a + b + c$, this problem is directly related to the problem of finding an asymptotic lower bound for the competitive ratio of our problem. Thus, the value of this formula is also bounded by ϕ^5.

In the above lower bound example, there are only two different kinds of processing sets. Thus, this example can serve a lower bound example for GoS eligibility with two GoS levels.

Corollary 4 *Any online algorithm for* $P \mid r_i, \mathcal{M}_i(2GoS(k)), pmtn \mid C_{\max}$ *has a competitive ratio at least* $1 + \phi^5 \approx 1.0917$*, where* $\phi = \frac{\sqrt{5}-1}{2}$.

Now we restrict ourselves to the case of two machines.

Lemma 5 *The lower bound of the competitive ratio of online scheduling problems with two machines subject to eligibility constraints are as follows:*

(*i*) *Any online algorithm for* $P2 \mid r_i, \mathcal{M}_i \mid C_{\max}$ *has a competitive ratio at least* 2.
(*ii*) *Any online algorithm for* $P2 \mid r_i, \mathcal{M}_i, restart \mid C_{\max}$ *has a competitive ratio at least* 1.5687.

Proof We prove lower bounds of the competitive ratio in each problem by an adversary argument.

(*i*) At time $t = 0$, job J_1 arrives with $p_1 = 1$ and $\mathcal{M}_1 = \{M_1, M_2\}$. By a symmetry of machines, we may assume that job J_1 is assigned to machine M_1 at time S_1. If $S_1 \geq 1$, then no job arrives later. Then $C_{\max}(\sigma)/C_{\max}(\pi) \geq 2$. Otherwise, job J_2 arrives with $p_2 = 1 - S_1$ and $\mathcal{M}_2 = \{M_1\}$ at time $t = S_1 + \varepsilon$. Then the completion time of job J_2 is $S_1 + 1 + (1 - S_1) = 2$. In this case the optimum makespan is $1 + \varepsilon$. Thus, $C_{\max}(\sigma)/C_{\max}(\pi) = 2/(1 + \varepsilon)$. When ε goes to 0, the competitive ratio approaches 2.

(*ii*) At time $t = 0$, job J_1 arrives with $p_1 = 1$ and $\mathcal{M}_1 = \{M_1, M_2\}$. Since this model admits restarts and machines M_1 and M_2 have a symmetry, without loss of generality, we may assume that job J_1 is assigned to machine M_1 at time $t = 0$. Job J_2 with $p_2 = 0.1374$ and $\mathcal{M}_2 = \{M_1\}$ arrives at time $r_2 = 0.4312$.

If we assign job J_2 to machine M_1 at its arrival time and job J_1 restarts at machine M_2, then we get job J_3 with $p_3 = 0.3531$ and $\mathcal{M}_3 = \{M_2\}$ at time $r_3 = 0.7844$. In the current optimal schedule, job J_1 is processed on machine M_1 from time 0 to time 1, job J_2 is processed on machine M_1 from time 1 to time $1 + p_2$, and job J_3 is processed on machine M_2 from time r_3 to time $r_3 + p_3$, implying that the current optimum makespan is $C_{\max}(\pi) = \max\{1 + p_2, r_3 + p_3\} = 1.1375$. If restart occurs, the makespan generated by the algorithm is at least $r_3 + 1 = 1.7844$. Otherwise, the makespan by the algorithm is at least $r_2 + 1 + p_3 = 1.7844$. Thus, in both cases, $C_{\max}(\sigma)/C_{\max}(\pi) \geq 1.5687$.

If we continue processing job J_1 instead of job J_2's start, then job J_3 arrives with $p_3 = 0.4313$ and $\mathcal{M}_3 = \{M_1\}$ at time $r_3 = 0.5687$. In the current optimal schedule, job J_1 is processed on machine M_2 from time 0 to time 1, job J_2 is processed on machine M_1 from time r_2 to time $r_2 + p_2$, and job J_3 is processed on machine M_1 from time r_3 to time $r_3 + p_3 = 1.0$, implying that the current optimum makespan is 1. If the restart of job J_1 occurs, the makespan by the algorithm is at least $r_3 + 1 = 1.5687$. Otherwise, the makespan generated by the algorithm is at least $p_1 + p_2 + p_3 = 1.5687$. Thus, $C_{\max}(\sigma)/C_{\max}(\pi) \geq 1.5687$ in both cases. □

Theorem 3 *Any online algorithm without delay for* $P2 \mid r_i, \mathcal{M}_i \mid C_{\max}$ *is optimal with a competitive ratio of 2.*

Proof Consider an online algorithm not allowing delay. Then the makespan by the algorithm is at most $\sum_{J_i \in J} p_i$. Since the optimal makespan is at least $\sum_{J_i \in J} p_i/2$, Lemma 5 (*i*) completes the proof. □

Note that we have already proved that any online algorithm for $P2 \mid r_i, \mathcal{M}_i, pmtn \mid C_{\max}$ has a competitive ratio of at least $1 + 1/8 = 1.125$ in Lemma 3.

Lemma 6 *The lower bounds of the competitive ratios of two machine scheduling problems subject to GoS eligibility constraints are the following:*

(i) *Any online algorithm for* $P2 \mid r_i, \mathcal{M}_i(GoS) \mid C_{\max}$ *has a competitive ratio at least* $1 + \lambda \approx 1.5550$*, where* λ *is a solution of* $\lambda^3 - 2\lambda^2 - \lambda + 1 = 0$ *in the range* $(0.5, 0.6)$.

(ii) *Any online algorithm for* $P2 \mid r_i, \mathcal{M}_i(GoS), restart \mid C_{\max}$ *has a competitive ratio at least* $4/3$.

Proof We show these results through a class of adversary examples.

(i) At time $t = 0$, job J_1 arrives with $p_1 = 1$ and $\mathcal{M}_1 = \{M_1, M_2\}$. If job J_1 is assigned to machine M_1, the competitive ratio can be up to 2 by using the lower bound example of $P2 \mid r_i, \mathcal{M}_i \mid C_{\max}$. Thus, we assume that job J_1 is assigned to machine M_2 at time S_1. Note that the current optimal makespan is 1. If $S_1 \geq 0.5550$, then no additional job arrives and $C_{\max}(\sigma)/C_{\max}(\pi) \geq 1.5550$.

Now we consider the case of $S_1 < 0.5550$. Job J_2 arrives at time $r_2 = S_1$ with $p_2 = 1.2470$ and $\mathcal{M}_2 = \{M_1, M_2\}$. If the starting time of job J_2, S_2, is greater than or equal to $S_1 + 1$ then $C_{\max}(\sigma) = S_2 + p_2 \geq S_1 + 1 + p_2$. In the current optimal schedule job J_1 is processed on machine M_2 and job J_2 is processed on machine M_1, implying that $C_{\max}(\pi) = r_2 + p_2 = S_1 + p_2$. Thus, we have

$$\frac{C_{\max}(\sigma)}{C_{\max}(\pi)} \geq \frac{S_1 + 1 + p_2}{S_1 + p_2} = 1 + \frac{1}{S_1 + p_2} \geq 1.5550.$$

Otherwise, $S_2 < S_1 + 1$ implying that job J_2 must be assigned to machine M_1. Then job J_3 arrives at time $r_3 = S_2$ with $p_3 = 1 + p_2 - S_2$ and $\mathcal{M}_3 = \{M_1\}$. So, $C_{\max}(\sigma) = S_2 + p_2 + (1 + p_2 - S_2) = 1 + 2p_2$. In the current optimal schedule, jobs 1 and 2 are processed on machine M_2 while job J_3 starts on machine M_1 at its arrival time, implying that the current optimal makespan is $C_{\max}(\pi) = \max\{1 + p_2, r_3 + p_3\} = 1 + p_2$. Thus,

$$\frac{C_{\max}(\sigma)}{C_{\max}(\pi)} \geq \frac{1 + 2p_2}{1 + p_2} = 1 + \frac{p_2}{1 + p_2} = 1.5550.$$

(ii) At time $t = 0$, two jobs, job J_1 with $p_1 = 1$, and $\mathcal{M}_1 = \{M_1, M_2\}$ and job J_2 with $p_2 = 2$ and $\mathcal{M}_2 = \{M_1, M_2\}$ arrive. Then the current optimal makespan is 2.

Consider the status of job J_2 at time 1. If job J_2 is not running on either machine M_1 or machine M_2 at time 1, job J_2 must start (or restart) after time 1 and be completed after time 3. It means that $C_{\max}(\sigma)/C_{\max}(\pi) \geq 3/2 > 4/3$. If job J_2 is running on machine M_1 at time 1, then job J_3 arrives with $p_3 = 1$ and $\mathcal{M}_3 = \{M_1\}$. In the current optimal schedule, jobs J_1 and J_3 start on machine M_1 at their arrival times and job J_2 starts on machine M_2 at time 0, implying that $C_{\max}(\pi) = 2$. If job J_2 restarts, then $C_{\max}(\sigma) \geq 3$. Otherwise job J_3 can start after job J_2 and it implies $C_{\max}(\sigma) \geq 3$. Thus, $C_{\max}(\sigma)/C_{\max}(\pi) \geq 3/2 > 4/3$.

Now consider the case where job J_2 is running on machine M_2 at time 1. Then job J_3 arrives with $p_3 = 2$ and $\mathcal{M}_3 = \{M_1, M_2\}$. Consider the status of machine M_1 at time $t = 2$.

Case 1 If job J_3 is running on machine M_1, then job J_4 arrives at time $t = 2$ with $p_4 = 1$ and $\mathcal{M}_4 = \{M_1\}$. In the current optimal schedule, jobs J_1 and J_3 start on

machine M_2 at their arrival times and jobs J_2 and J_4 start on machine M_1 at their arrival times, implying that $C_{\max}(\pi) = 3$. If job J_3 restarts then $C_{\max}(\sigma) \geq 4$. Otherwise job J_4 can start after job J_3 and it implies $C_{\max}(\sigma) \geq 4$. Thus, $C_{\max}(\sigma)/C_{\max}(\pi) \geq 4/3$ in both cases.

Case 2 If job J_2 is running on machine M_1, this implies that job J_2 restarts after time $t = 1$. Note that job J_2 was running on machine M_2 at time $t = 1$. Then, job J_4 arrives at time $t = 2$ with $p_4 = 1$ and $\mathcal{M}_4 = \{M_1\}$. In the current optimal schedule, jobs J_1 and J_3 start on machine M_2 at their arrival times and jobs J_2 and J_4 start on machine M_1 at their arrival times, implying that $C_{\max}(\pi) = 3$. Since jobs J_2 and J_3 start after time $t = 1$, $C_{\max}(\sigma) \geq 4$. Thus, $C_{\max}(\sigma)/C_{\max}(\pi) \geq 4/3$.

Case 3 If no jobs are running on machine M_1, then there are two possibilities. If job J_3 is running on machine M_2 at time $t = 2$, this implies that job J_2 restarts after time $t = 1$. By the same argument as in *Case 2*, $C_{\max}(\sigma)/C_{\max}(\pi) \geq 4/3$. Otherwise, job J_3 must start (or restart) after time $t = 2$. Then no more jobs arrive later and $C_{\max}(\sigma)/C_{\max}(\pi) \geq 4/3$. □

Now we consider online problem $P2 \mid r_i, \mathcal{M}_i(GoS), pmtn \mid C_{\max}$ and its optimal algorithm. First we consider an offline algorithm for offline problem $P2 \mid \mathcal{M}_i(GoS), pmtn \mid C_{\max}$. Let L_j denote the total processing requirement of the jobs whose grade are j, i.e. $L_j = \sum_{i:g_i=j} p_i$. Let $p_{\max}$ denote the largest processing requirement of all jobs, i.e., $p_{\max} = \max_{i \in J}\{p_i\}$. Then $L = \max\{p_{\max}, L_1, (L_1 + L_2)/2\}$ is the optimal makespan. An optimal schedule is constructed in the following way: (1) schedule all jobs with grade 1 on machine M_1 contiguously from time zero to time L_1; (2) schedule jobs with grade 2 on machine M_1 from time L_1 to L without idle times and a running job at time L is preempted; (3) schedule the remaining jobs (with grade 2) on machine M_2 from time 0. In other words, we assign the least flexible jobs first and then apply McNaughton rule (1959) to the most flexible jobs. Let this algorithm be called LF–Mc. For online problem $P2 \mid r_i, \mathcal{M}_i(GoS), pmtn \mid C_{\max}$, we devise an algorithm using algorithm LF–Mc as a submodule.

Let J^0 be the set of jobs available at time 0. The scheduler applies algorithm LF–Mc and schedules the jobs in J^0, finishing at time f_0. The next arrival of a group of jobs occurs at time a_1. If $a_1 \geq f_0$, then the previous jobs are completed at time f_0 or before. Otherwise, if $a_1 < f_0$, the scheduler immediately preempts all running jobs. Let J^1 be the set of jobs to be scheduled at time a_1 which consists of jobs released at time a_1, the unscheduled jobs and preempted jobs. The scheduler now, at time a_1, applies algorithm LF–Mc to schedule jobs in J^1, starting at time a_1 and finishing at time f_1. The next arrival of a group of jobs occurs at time a_2. In general, let J^i be the set of jobs that are to be scheduled at time a_i. At time a_i, the scheduler uses algorithm LF–Mc to schedule the jobs in J^i from time a_i. Let f_i be the point in time when the schedule for J^i completes. Let this online algorithm be called algorithm on-LF–Mc. We can easily observe that the schedule generated by algorithm on-LF–Mc processes jobs with grade 1 as much as possible at any time point among all feasible schedules.

Theorem 4 *Algorithm on-LF–Mc for* $P2 \mid r_i, \mathcal{M}_i(GoS), pmtn \mid C_{\max}$ *has the optimal competitive ratio of 1.*

Proof We prove the theorem by contradiction. Let a_h be the last arrival time. Suppose the thesis is not true. Then, there must be a problem instance, called a counterexample,

with which algorithm on-LF–Mc does not construct an optimal schedule. If there exists a counterexample, there must be a counterexample with the minimum number of distinct arrival times, called a minimum counterexample. Among all minimum counterexamples, there exists a minimum counterexample without idle time period before a_h. If there exists an idle time period in $[a_\nu, a_{\nu+1}]$, then we can add a job such that its processing requirement is the same as the length of the idle time period, it is released at time a_ν and it is eligible to both machines. This addition does not increase the number of distinct arrival times. From now on, we only consider a minimum counterexample without idle times before time a_h.

For the case with $h = 0$, the theorem holds since algorithm LF–Mc is optimal for the offline problem. By definition of the minimum counterexample, for $\nu \leq h - 1$, all jobs that are released at time a_ν or before are scheduled optimally. In other words, when we only consider jobs that arrive at time a_ν or before, the current optimal makespan is f_ν. In case $a_h \geq f_{h-1}$, it is easy to show that the schedule by on-LF-MC is optimal. Thus, we consider the case for $a_h < f_{h-1}$. For convenience, let L^h, $p^h_{\max}$, L^h_1 and L^h_2 denote L, $p_{\max}$, L_1 and L_2 with set J^h, respectively.

We consider three possible cases of the schedule on f_h. Obviously $f_h = a_h + L^h$.

Case 1 Suppose that $f_h = a_h + p^h_{\max}$. If a job with processing requirement of $p_{\max}$ arrives at time a_h, then the optimal makespan is at least $a_h + p^h_{\max}$. Otherwise, if the job arrives before time a_h, then in the previous schedule at time a_h its remaining processing requirement is $p^h_{\max}$, implying that $f_{h-1} = f_h$. Since the optimal makespan is at least f_{h-1}, the schedule is optimal, which is a contradiction.
Case 2 Suppose that $f_h = a_h + L^h_1$. Algorithm on-LF–Mc processes jobs with grade 1 as much as possible up to time a_h and thus $a_h + L^h_1$ is the lower bound of the optimal makespan. Thus, it is a contradiction.
Case 3 Suppose that $f_h = a_h + (L^h_1 + L^h_2)/2$. Since there are no idle times before time a_h, the total processing requirement is $2a_h + (L^h_1 + L^h_2)$, implying that the optimal makespan is at least $a_h + (L^h_1 + L^h_2)/2$. Thus, f_h is the optimal makespan, which is a contradiction. □

Table 2 summarizes the results on competitive ratios and their lower bounds of the problems under consideration.

4.2 Online service scheduling

Online service scheduling was first introduced by Wang et al. (2009). In online scheduling over time, we have a pool of unscheduled jobs and we may select a job among the pool to assign to an idle machine. The pool of unscheduled jobs changes over time according to the assignment of jobs and the arrival of new jobs. Customers (jobs) are classified as either 'ordinary' or 'special'. Ordinary customers can be served on any service facility (machines), while special customers can be served only on a flexible service facility. Formally, there are m machines, of which the first k ($1 \leq k \leq m$) machines are flexible and can process both ordinary and special jobs while the remaining $m - k$ machines can process only ordinary jobs. Customers arrive over time and

their needs become known upon arrival. Any service, once started, will be carried out to completion. The objective is to minimize the makespan.

The only difference lies in the fact that in online service scheduling there is a restriction with regard to the order of assignment that is related to the order of arrivals. Typically, among jobs in the same class, the order of assignment is consistent with the order of arrival. We only take into account online algorithms satisfying such conditions even though there are more efficient algorithms in terms of the competitive ratio. Thus, while the aim of the traditional online scheduling research is to find an online algorithm producing a schedule with a smaller competitive ratio, this study focuses on the analysis of competitive ratios of practical service policies.

Wang et al. (2009) and Wang and Xing (2010) investigate three categories of service policies used in practice;

(i) priority policies,
(ii) non-priority policies,
(iii) mixed policies.

(*i*) *Priority policies.* If, in a service system, customers requiring special services are considered more important and have to receive preferential treatment, policies with priorities are typically established. One such policy is the 'Designated Flexible Servers' (DFS) policy. In such a policy, special customers have priority over ordinary customers. The flexible servers serve special customers in the order of their arrivals and, when all special customers in the queue have been served, the flexible servers will remain idle and have to wait for the arrival of the next special customer, even when there are ordinary customers in the queue. Another policy is the 'Special Customers First' (SCF) policy. Under such a policy, the flexible servers first serve special customers in the order of their arrivals. In the absence of special customers, they will serve the first ordinary customer in the queue.

(*ii*) *Non-priority policies.* If no priorities are considered, two policies come into consideration. The easiest and most impartial one is the 'First-Come, First-Served' (FCFS) policy. All customers are in the same queue in order of their arrivals and 'queue jumping' is not allowed. Note that in the FCFS policy, if the first customer in the queue is special and all flexible servers are busy, then all customers in the queue must wait even if some dedicated servers are idle. Therefore, although the FCFS policy is absolutely impartial, it may waste service resources. As a result, the 'First-Fit, First-Served' (FFFS) policy may be adopted. In such a policy, the first idle server serves the first customer that it is able to serve.

(*iii*) *Mixed policies.* Consideration of both prioritization and impartiality may lead to mixed service policies. We consider two such policies: (1) Set aside some of the flexible servers to serve customers according to the DFS policy and apply the FFFS policy to all other servers. Such a policy will be called the Mixed policy with DFS (MDFS). (2) Set aside some of the flexible servers to serve customers according to the SCF policy and apply the FFFS policy to all other servers. Such policy will be called the Mixed policy with SCF (MSCF).

Let $\rho(H)$ denote the competitive ratio of policy H for $H \in \{\text{DFS, SCF, FCFS, FFFS}\}$ and let $\rho(H'(k_1))$ denote the competitive ratio of the mixed policy H' for $H' \in \{\text{MDFS, MSCF}\}$, where the first k machines are flexible among the m machines

and in mixed policies, the number of flexible machines to which the DFS or SCF policy is applied is k' while the number of flexible machines to which the FFFS policy is applied is $k - k'$ for $0 \le k_1 \le k$. Then the following results can be obtained for priority policies:

$$\rho(\text{DFS}) = 1 + \frac{m-1}{m-k}$$
$$\rho(\text{SCF}) = \begin{cases} 3 - \frac{1}{k} & \text{for} \quad 1 \le k \le \frac{m}{2}, \\ 1 + \frac{m-1}{k} & \text{for} \quad \frac{m}{2} < k \le m; \end{cases}$$

for non-priority policies:

$$\rho(\text{FCFS}) = 1 + \frac{m-1}{k}$$
$$\rho(\text{FFFS}) = \begin{cases} 4 - \frac{2k}{m} - \frac{1}{k} & \text{for} \quad 1 \le k \le \frac{m}{2}, \\ 1 + \frac{m-1}{k} & \text{for} \quad \frac{m}{2} < k \le m; \end{cases}$$

for mixed policies:

$$\rho(\text{MDFS}(k_1))$$
$$= \begin{cases} 2 + 2\frac{(k-k_1)(m-k)}{k(m-k_1)} - \frac{1}{k} & \text{for} \quad 1 \le k \le \frac{m}{2} \quad \text{and} \quad 0 \le k_1 \le \frac{m(k-1)}{2m-k-1}, \\ 1 + \frac{k-1}{k_1} & \text{for} \quad 1 \le k \le \frac{m}{2} \quad \text{and} \quad \frac{m(k-1)}{2m-k-1} \le k_1 \le \frac{m(k-1)}{m+k-2}, \\ 1 + \frac{m-1}{m-k_1} & \text{for} \quad 1 \le k \le \frac{m}{2} \quad \text{and} \quad \frac{m(k-1)}{m+k-2} \le k_1 \le k \\ 1 + \frac{m-1}{k} & \text{for} \quad \frac{m}{2} < k \le \frac{2m}{3} \quad \text{and} \quad 0 \le k_1 \le 2k - m, \\ 2 + 2\frac{(k-k_1)(m-k)}{k(m-k_1)} - \frac{1}{k} & \text{for} \quad \frac{m}{2} < k \le \frac{2m}{3} \quad \text{and} \quad 2k - m \le k_1 \le \frac{(2k-1)(m-k)}{2m-k-1}, \\ 1 + \frac{m-1}{m-k_1} & \text{for} \quad \frac{m}{2} < k \le \frac{2m}{3} \quad \text{and} \quad \frac{(2k-1)(m-k)}{2m-k-1} \le k_1 \le k, \\ 1 + \frac{m-1}{k} & \text{for} \quad \frac{2m}{3} < k \le m \quad \text{and} \quad 0 \le k_1 \le m - k, \\ 1 + \frac{m-1}{m-k_1} & \text{for} \quad \frac{2m}{3} < k \le m \quad \text{and} \quad m - k \le k_1 \le k. \end{cases}$$
$$\rho(\text{MSCF}(k_1)) = \begin{cases} 3 + 2\frac{(k-k_1)(m-2k)}{k(m-k_1)} - \frac{1}{k} & \text{for} \quad 1 \le k \le \frac{m}{2}, \\ 1 + \frac{m-1}{k} & \text{for} \quad \frac{m}{2} < k \le m. \end{cases}$$

From the expressions above we have the following chains of inequalities. where the competitive ratio $\rho(\text{MDFS}(k_1))$ of the MDFS policy is minimized at $k_1 = k^*$.

$1 \le k \le m/2$						
Policy	Mixed	Priority	Priority	Mixed	Non-priority	Non-priority
ρ	$\rho(\text{MDFS}(k^*))$	$\le \rho(\text{DFS})$	$\le \rho(\text{SCF})$	$\le \rho(\text{MSCF}(k_1))$	$\le \rho(\text{FFFS})$	$\le \rho(\text{FCFS})$
$m/2 < k < m$						
Policy	Mixed	Mixed	Non-priority	Non-priority	Priority	Priority
ρ	$\rho(\text{MDFS}(k^*))$	$\le \rho(\text{MSCF}(k_1))$	$= \rho(\text{FFFS})$	$= \rho(\text{FCFS})$	$= \rho(\text{SCF})$	$\le \rho(\text{DFS})$

4.3 The weighted completion and flow time

The total completion time ($\sum C_i$) and the total weighted completion time ($\sum w_i C_i$) are important objectives in scheduling theory and have received much attention. Related objectives are the total flow time ($\sum F_i$) and the total weighted flow time ($\sum w_i F_i$) where the flow time is defined as the completion time minus the release time, that is $F_i = C_i - r_i$. Since the total (weighted) release time is a constant, the total (weighted) completion time problem and the total (weighted) flow time problem have the same optimal schedules. However, the approximation algorithms have quite different competitive ratios because the optimal objective function values are different for each problem.

Hall et al. (1997) prove that the online problem $R \mid r_i \mid \sum w_i C_i$ has an 8-competitive algorithm. Since $P \mid r_i, \mathcal{M}_i \mid \sum w_i C_i$ is a special case of $R \mid r_i \mid \sum w_i C_i$, problem $P \mid r_i, \mathcal{M}_i \mid \sum w_i C_i$ also has an 8-competitive algorithm.

However, for the problem of minimizing the total flow time, there does not exist an online algorithm with a bounded competitive ratio even for the case where all jobs have unit processing requirements, that is even for $P \mid r_i, \mathcal{M}_i, p_i = 1 \mid \sum F_i$ by Garg and Kumar (2007). Also, even for the problem with two machines subject to GoS eligibility, denoted by $P2 \mid r_i, \mathcal{M}_i(GoS) \mid \sum F_i$, the competitive ratio can be arbitrarily large.

5 Future research

In this paper we presented the state of the art in online scheduling subject to machine eligibility constraints and with the makespan as objective. We considered both online scheduling paradigms, namely online over list and online over time. A fair amount of research has been done on this topic; Tables 1 and 2 provide comprehensive overviews displaying all the results in online over list and online over time, respectively. There are still many open problems remaining in both paradigms.

In online over list the following problems seem to be of interest.

Problem 1. What are the most general conditions on a parallel machine scheduling environment under which Azar's algorithm remains optimal?

Problem 2. What is the best possible algorithm for interval eligibility constraints?

Problem 3. What are the exact competitive ratios for the problem subject to GoS eligibility constraints?

Problem 4. How does the value of partial information in a semi-online scheduling problem depend on the type of eligibility constraints?

There are many open problems in the online over time paradigm as well.

Problem 5. What is the exact competitive ratio when an optimal or approximation offline algorithm is repeatedly applied at each arrival of a job to a problem subject to eligibility constraints?

Problem 6. How should one analyze algorithms for nonclairvoyant online scheduling problems subject to eligibility constraints?

Problem 7. Does the restart option provide any advantage when minimizing the makespan subject to eligibility constraints?

Table 1 Lower and upper bounds of competitive ratios in online over list

Problem	$\mathcal{L}(\rho)$	References	$\mathcal{A}$	$\rho(\mathcal{A})$	References	Gap
$P \mid \mathcal{M}_i \mid C_{\max}$	$\lceil \log_2(m+1) \rceil$	Azar et al. (1995)	$\mathcal{AW}$	$\lceil \log_2 m \rceil + 1$	Azar et al. (1995)	≤ 1
$P \mid \mathcal{M}_i \mid C_{\max}$	–	–	$\mathcal{AW}$	$\log_2 m + 1$	Hwang et al. (2004a)	≤ 1
$P \mid \mathcal{M}_i \mid C_{\max}$	$LB(m)^*$	Lim et al. (2010)	$\mathcal{AW}$	$k^\diamond + m/2^k$	Lim et al. (2010)	≤ 0.1967
$P \mid \mathcal{M}_i, semi(sum, max, opt) \mid C_{\max}$	$LB(m)^*$	Lim et al. (2010)	$\mathcal{AW}$	$k + m/2^k$	Lim et al. (2010)	≤ 0.1967
$P2 \mid \mathcal{M}_i \mid C_{\max}$	2	Azar et al. (1995)	$\mathcal{AW}$	2	Azar et al. (1995)	0
$P3 \mid \mathcal{M}_i \mid C_{\max}$	2.5	Lim et al. (2010)	$\mathcal{AW}$	2.5	Lim et al. (2010)	0
$P4 \mid \mathcal{M}_i \mid C_{\max}$	3	Azar et al. (1995)	$\mathcal{AW}$	3	Azar et al. (1995)	0
$P5 \mid \mathcal{M}_i \mid C_{\max}$	3.25	Lim et al. (2010)	$\mathcal{AW}$	3.25	Lim et al. (2010)	0
$P6 \mid \mathcal{M}_i \mid C_{\max}$	3.5	Lim et al. (2010)	$\mathcal{AW}$	3.5	Lim et al. (2010)	0
$P7 \mid \mathcal{M}_i \mid C_{\max}$	$3.69\dot{4}$	Lim et al. (2010)	$\mathcal{AW}$	3.7	Lim et al. (2010)	$0.0055\dot{6}$
$P \mid \mathcal{M}_i(tree) \mid C_{\max}$	e	Bar-Noy et al. (2001)	Doubling	5	Bar-Noy et al. (2001)	$5 - e$
$P \mid \mathcal{M}_i(tree), p_i = 1 \mid C_{\max}$	e	Bar-Noy et al. (2001)	Doubling	4	Bar-Noy et al. (2001)	$4 - e$
$P \mid \mathcal{M}_i(tree), pmtn \mid C_{\max}$	e	Bar-Noy et al. (2001)	Doubling	4	Bar-Noy et al. (2001)	$4 - e$
$P \mid \mathcal{M}_i(GoS) \mid C_{\max}$	e	Bar-Noy et al. (2001)	IntegralHarmonic	$e + 1$	Bar-Noy et al. (2001)	1
$P \mid \mathcal{M}_i(GoS), p_i = 1 \mid C_{\max}$	e	Bar-Noy et al. (2001)	IntegralHarmonic	e	Bar-Noy et al. (2001)	0
$P \mid \mathcal{M}_i(GoS), pmtn \mid C_{\max}$	e	Bar-Noy et al. (2001)	IntegralHarmonic	e	Bar-Noy et al. (2001)	0
$P2 \mid \mathcal{M}_i(GoS) \mid C_{\max}$	5/3	Park et al. (2006), Jiang et al. (2006)	Online, H1	5/3	Park et al. (2006), Jiang et al. (2006)	0
$P3 \mid \mathcal{M}_i(GoS) \mid C_{\max}$	2	Tan and Zhang (2010b)	HT	2	Tan and Zhang (2010b)	0
$P4 \mid \mathcal{M}_i(GoS) \mid C_{\max}$	2	Tan and Zhang (2010b), Observation 1	HT	2.333	Tan and Zhang (2010b)	0.333
$P5 \mid \mathcal{M}_i(GoS) \mid C_{\max}$	2	Tan and Zhang (2010b), Observation 1	HT	2.610	Tan and Zhang (2010b)	0.610
$P3 \mid \mathcal{M}_i(GoS), pmtn \mid C_{\max}$	3/2	Dosa and Epstein (2008)	Algorithm 3	3/2	Dosa and Epstein (2008)	0

Table 1 continued

Problem	$\mathcal{L}(\rho)$	References	$\mathcal{A}$	$\rho(\mathcal{A})$	References	Gap
$P2 \mid \mathcal{M}_i(GoS), semi(sum) \mid C_{\max}$	3/2	Park et al. (2006)	Semi-Online	3/2	Park et al. (2006)	0
$P2 \mid \mathcal{M}_i(GoS), semi(opt) \mid C_{\max}$	3/2	Wu and Yang (2010)	Semi-Online	3/2	Wu and Yang (2010)	0
$P2 \mid \mathcal{M}_i(GoS), semi(max) \mid C_{\max}$	$(\sqrt{5}+1)/2$	Wu and Yang (2010)	Semi-Online	$(\sqrt{5}+1)/2$	Wu and Yang (2010)	0
$P2 \mid \begin{matrix}\mathcal{M}_i(GoS),\\ semi(p_i \in [a, \alpha a])\end{matrix} \mid C_{\max}$	$\frac{1+\alpha}{2},\ 1 < \alpha < 2$ $\frac{3}{2},\ 2 \le \alpha < 5,$ $\frac{4+\alpha}{6},\ 5 \le \alpha < 6$	Liu et al. (2010)	B-Online	$\frac{1+\alpha}{2},\ \frac{25}{14} < \alpha < 2$ $\frac{3}{2},\ 2 \le \alpha < 5,$ $\frac{4+\alpha}{6},\ 5 \le \alpha < 6^{**}$	Liu et al. (2010)	
$P2 \mid \begin{matrix}\mathcal{M}_i(GoS),\\ semi(Sum,\ p_i \in [a, \alpha a])\end{matrix} \mid C_{\max}$	$\frac{1+\alpha}{2}, 1 < \alpha < 2$	Liu et al. (2010)	B-Sum-Online	$\frac{1+\alpha}{2},\ 1 < \alpha < 2^{***}$	Liu et al. (2010)	
$P \mid \mathcal{M}_i(2GoS(k)) \mid C_{\max}$	2	Jiang (2008)	HA	$(12+4\sqrt{2})/7$	Jiang (2008)	0.522
$P \mid \mathcal{M}_i(2GoS(k)) \mid C_{\max}$	–	–	TLS	$1+\frac{m^2-m}{m^2-km+k^2} < \frac{7}{3}$	Zhang et al. (2009)	0.333
$P3 \mid \mathcal{M}_i(2GoS(1)) \mid C_{\max}$	1.824	Zhang et al. (2009)	TLS	1.857	Zhang et al. (2009)	0.023
$P3 \mid \mathcal{M}_i(2GoS(2)) \mid C_{\max}$	1.801	Zhang et al. (2009)	TLS	1.857	Zhang et al. (2009)	0.056
$P4 \mid \mathcal{M}_i(2GoS(1)) \mid C_{\max}$	1.848	Zhang et al. (2009)	TLS	1.923	Zhang et al. (2009)	0.075
$P4 \mid \mathcal{M}_i(2GoS(2)) \mid C_{\max}$	1.907	Zhang et al. (2009)	SLS	2	Zhang et al. (2009)	0.093
$P5 \mid \mathcal{M}_i(2GoS(1)) \mid C_{\max}$	1.848	Zhang et al. (2009)	TLS	1.952	Zhang et al. (2009)	0.104
$P5 \mid \mathcal{M}_i(2GoS(2)) \mid C_{\max}$	1.907	Zhang et al. (2009)	SLS	2	Zhang et al. (2009)	0.093
$P6 \mid \mathcal{M}_i(2GoS(1)) \mid C_{\max}$	1.829	Zhang et al. (2009)	TLS	1.968	Zhang et al. (2009)	0.139
$P6 \mid \mathcal{M}_i(2GoS(2)) \mid C_{\max}$	1.907	Zhang et al. (2009)	SLS	2	Zhang et al. (2009)	0.093
$P \mid \mathcal{M}_i(interval) \mid C_{\max}$	$\frac{1}{2}\log_2 m$	Bar-Noy et al. (2001)	$\mathcal{AW}$	$1+\log_2 m$	Azar et al. (1995)	$1+\frac{1}{2}\log_2 m$

Table 1 continued

Problem	$\mathcal{L}(\rho)$	References	$\mathcal{A}$	$\rho(\mathcal{A})$	References	Gap
$Q2 \mid \mathcal{M}_i, \mid C_{\max}$	$1+s^{\dagger}$, $1+1/s^{\ddagger}$	Lee et al. (2009)	HSF	$1+s^{\dagger}$, $1+1/s^{\ddagger}$	Lee et al. (2009)	0
$Q2 \mid \mathcal{M}_i(GoS) \mid C_{\max}$	$\min\left\{1+s, 1+\frac{1+s}{1+s+s^2}\right\}^{\dagger}$, $\min\left\{1+1/s, 1+\frac{2s}{1+s+s^2}\right\}^{\ddagger}$	Tan and Zhang (2010a)	Algorithm A1†	$\min\left\{1+s, 1+\frac{1+s}{1+s+s^2}\right\}^{\dagger}$,	Tan and Zhang (2010a)	0
			Algorithm A2‡	$\min\left\{1+1/s, 1+\frac{2s}{1+s+s^2}\right\}^{\ddagger}$	Lee et al. (2009)	
$Q2 \mid \mathcal{M}_i(GoS), pmtn \mid C_{\max}$	$\frac{(s+1)^2}{s^2+s+1}^{\dagger}$, $\frac{s(s+1)^2}{s^3+s^2+1}^{\ddagger}$	Dosa and Epstein (2008)	Algorithm 2†	$\frac{(s+1)^2}{s^2+s+1}^{\dagger}$,	Dosa and Epstein (2008)	0
			Algorithm 1‡	$\frac{s(s+1)^2}{s^3+s^2+1}^{\ddagger}$		

$^{\diamond} k = \lfloor \log_2 m \rfloor$

$^{*} LB(m) = \begin{cases} 1 & m = 1 \\ LB(\gamma_m) + \lceil \gamma_m/(m-\gamma_m) \rceil^{-1} & m \geq 2 \end{cases}$, where $\gamma_m = \arg\max_{\lceil m/2 \rceil \leq i \leq m-1} \left\{ LB(i) + \lceil i/(m-i) \rceil^{-1} \right\}$

** Only when the optimal makespan $\geq 20a$

*** Only when the total processing requirement of all jobs is at least $\frac{2\alpha a}{\alpha - 1}$

† For $0 < s < 1$, $s = v_1/v_2$

‡ For $s > 1$, $s = v_1/v_2$

Table 2 Lower and upper bounds of competitive ratios in online over time

Problem	$\mathcal{L}(\rho)$	References	$\mathcal{A}$	$\rho(\mathcal{A})$	References
$P \mid r_i \mid C_{\max}$	1.3473	Chen and Vestjens (1997)	LPT	1.5	Chen and Vestjens (1997)
$P \mid r_i, restart \mid C_{\max}$	$\sqrt{6}/2$	Lemma 1	LPT	1.5	Chen and Vestjens (1997), Observation 3
$P \mid r_i, pmtn \mid C_{\max}$	1	–	Alg. A	1	Hong and Leung (1992)
$P \mid r_i, \mathcal{M}_i \mid C_{\max}$	2	Lemma 5 (i)	LP + Rounding*	$4-\frac{2}{m}$	Corollary 1 (i), Shchepin and Vakhania (2005)**
$P \mid r_i, \mathcal{M}_i, restart \mid C_{\max}$	1.5687	Lemma 5 (ii), Observation 1	LP + Rounding*	$3-\frac{1}{m}$	Corollary 2 (i), Shchepin and Vakhania (2005)**
$P \mid r_i, \mathcal{M}_i, pmtn \mid C_{\max}$	5/4	Corollary 3	LP*	2	Corollary 1 (ii), Lawler and Labetoulle (1978)**
$P \mid r_i, \mathcal{M}_i(GoS) \mid C_{\max}$	1.5550	Lemma 6 (i), Observation 1	PTAS*	2+ϵ	Corollary 1 (iii), Ou et al. (2008)**
$P \mid r_i, \mathcal{M}_i(GoS), restart \mid C_{\max}$	4/3	Lemma 6 (ii), Observation 1	PTAS*	2+ϵ	Corollary 2 (ii), Ou et al. (2008)**
$P \mid r_i, \mathcal{M}_i(GoS), pmtn \mid C_{\max}$	1.0917	Lemma 4	LP*	2	Corollary 1 (ii), Lawler and Labetoulle (1978)**
$P \mid r_i, \mathcal{M}_i(nested) \mid C_{\max}$	1.5550	Lemma 6 (i), Observation 1,2	PTAS*	2+ϵ	Corollary 1 (iv), Muratore et al. (2010)**
$P \mid r_i, \mathcal{M}_i(nested), restart \mid C_{\max}$	4/3	Lemma 6 (ii), Observation 1,2	PTAS*	2+ϵ	Corollary 2 (iii), Muratore et al. (2010)**
$P \mid r_i, \mathcal{M}_i(nested), pmtn \mid C_{\max}$	1.148	Lemma 3	LP*	2	Corollary 1 (ii), Lawler and Labetoulle (1978)**
$P \mid r_i, \mathcal{M}_i(tree) \mid C_{\max}$	1.5550	Lemma 6 (i), Observation 1,2	TreeFeasibility*	8/3	Corollary 1 (v), Huo and Leung (2010b)**
$P \mid r_i, \mathcal{M}_i(tree), restart \mid C_{\max}$	4/3	Lemma 6 (ii), Observation 1,2	TreeFeasibility*	7/3	Corollary 2 (iv), Huo and Leung (2010b)**
$P2 \mid r_i \mid C_{\max}$	1.3820	Chen and Vestjens (1997)	SLEEPY	1.3820	Noga and Seiden (2001)
$P2 \mid r_i, restart \mid C_{\max}$	$\sqrt{6}/2$	Lemma 1	SLEEPY	1.3820	Noga and Seiden (2001), Observation 3
$P2 \mid r_i, pmtn \mid C_{\max}$	1	–	Alg. A	1	Hong and Leung (1992)
$P2 \mid r_i, \mathcal{M}_i \mid C_{\max}$	2	Lemma 5 (i)	arbitrary	2	Theorem 3
$P2 \mid r_i, \mathcal{M}_i, restart \mid C_{\max}$	1.5687	Lemma 5 (ii)	arbitrary	2	Theorem 3, Observation 3
$P2 \mid r_i, \mathcal{M}_i, pmtn \mid C_{\max}$	1.125	Lemma 3	arbitrary	2	Theorem 3, Observation 3
$P3 \mid r_i, \mathcal{M}_i, pmtn \mid C_{\max}$	1.148	Lemma 3, Observation 2	LP*	2	Corollary 1 (ii), Lawler and Labetoulle (1978)**

Table 2 continued

Problem	$\mathcal{L}(\rho)$	References	$\mathcal{A}$	$\rho(\mathcal{A})$	References
$Pm \mid r_i, \mathcal{M}_i, pmtn \mid C_{\max}$	$1 + \frac{(m-1)(m-2)}{2m(2m-3)}$ †	Lemma 2	LP*	2	Corollary 1 (ii), Lawler and Labetoulle (1978)**
$Pm \mid r_i, \mathcal{M}_i(nested), pmtn \mid C_{\max}$	1.148‡	Lemma 3	LP*	2	Corollary 1 (ii), Lawler and Labetoulle (1978)**
$P2 \mid r_i, \mathcal{M}_i(GoS) \mid C_{\max}$	1.5550	Lemma 6 (i)	arbitrary	2	Theorem 3, Observation 2
$P2 \mid r_i, \mathcal{M}_i(GoS), restart \mid C_{\max}$	4/3	Lemma 6 (ii)	arbitrary	2	Theorem 3, Observation 2, 3
$P2 \mid r_i, \mathcal{M}_i(GoS), pmtn \mid C_{\max}$	1	–	on-LF–Mc	1	Theorem 4
$P2 \mid r_i, \mathcal{M}_i, p_i = p \mid C_{\max}$	$(1+\sqrt{5})/2$	Lee et al. (2010a)	DLFJ	$(1+\sqrt{5})/2$	Lee et al. (2010a)
$P2 \mid r_i, \mathcal{M}_i(GoS), p_i = p \mid C_{\max}$	$\sqrt{2}$	Lee et al. (2010a)	SCD	$\sqrt{2}$	Lee et al. (2010a)

* The offline algorithm used as a subroutine
** The reference of the offline algorithm
† For $m \geq 4$
‡ For $m \geq 3$

There is still one major and challenging research problem:

Problem 8. Knowing the competitive ratio of a problem in one class (e.g., online over list), does that tell us anything about the competitive ratio of the corresponding problem in the other class (e.g., online over time)?

References

Albers S (2003) Online algorithms: a survey. Math Program Ser B 97:3–26

Azar Y, Naor J, Rom R (1995) The competitiveness of on-line assignments. J Algorithm 18:221–237

Bar-Noy A, Freund A, Naor J (2001) Online load balancing in a hierarchical server topology. SIAM J Comput 31:527–549

Chen B, Vestjens APA (1997) Scheduling on identical machines: how good is LPT in an on-line setting? Oper Res Lett 21:165–169

Dosa G, Epstein L (2008) Preemptive scheduling on a small number of hierarchical machines. Inf Comput 206(5):602–619

Garg N, Kumar A (2007) Minimizing average flow-time: upper and lower bounds. In: Proceedings of the 48th annual IEEE symposium on foundations of computer science, pp 603–613

Glass CA, Mills HR (2006) Scheduling unit length jobs with parallel nested machine processing set restrictions. Comput Oper Res 33:620–638

Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discret Math 5:287–326

Hall LA, Schulz AS, Shmoys DB, Wein J (1997) Scheduling to minimize average completion time: off-line and on-line approximation algorithms. Math Oper Res 22(3):513–544

Hong KS, Leung JY-T (1992) On-line scheduling of real-time tasks. IEEE Trans Comput 41:1326–1331

Huo Y, Leung JY-T (2010a) Parallel machine scheduling with nested processing set restrictions. Eur J Oper Res 204:229–236

Huo Y, Leung JY-T (2010b) Fast approximation algorithms for job scheduling with processing set restrictions, Theoretical Computer Science, to appear. doi:10.1016/j.tcs.2010.08.008

Hwang H-C, Chang SY, Hong Y (2004) A posterior competitiveness for list scheduling algorithm on machines with eligibility constraints. Asia-Pac J Oper Res 21:117–125

Hwang H-C, Chang SY, Lee K (2004) Parallel machine scheduling under a grade of service provision. Comput Oper Res 31:2055–2061

Jiang Y (2008) Online scheduling on parallel machines with two GoS levels. J Comb Optim 16:28–38

Jiang Y, He Y, Tang C (2006) Optimal online algorithms for scheduling on two identical machines under a grade of service. J Zhejiang Univ Sci A 7:309–314

Lawler EL, Labetoulle J (1978) On preemptive scheduling of unrelated parallel processors by linear programming. J ACM 25(4):612–619

Lee K, Leung JY-T, Pinedo M (2009) Online scheduling on two uniform machines subject to eligibility constraints. Theor Comput Sci 410:3975–3981

Lee K, Leung JY-T, Pinedo M (2010) Scheduling jobs with equal processing times subject to machine eligibility constraints. J Sched, to appear. doi:10.1007/s10951-010-0190-0

Lenstra JK, Shmoys DB, Tardos E (1990) Approximation algorithms for scheduling unrelated parallel machines. Math Program 46:259–271

Leung JY-T, Li C-L (2008) Scheduling with processing set restrictions: a survey. Int J Prod Econ 116:251–262

Lim K, Lee K, Chang SY (2010) On optimality of a greedy approach to the online scheduling under eligibility constraints, Working paper, Department of Industrial and Management Engineering, Pohang University of Science and Technology, Republic of Korea 790–784

Liu M, Xu Y, Chu C, Zheng F (2009) Online scheduling on two uniform machines to minimize the makespan. Theor Comput Sci 410(21–23):2099–2109

Liu M, Chu C, Xu Y, Zheng F (2010) Semi-online scheduling on 2 machines under a grade of service provision with bounded processing times. J Comb Optim, to appear. doi:10.1007/s10878-009-9231-z

Mandelbaum M, Shabtay D (2010) Scheduling unit length jobs on parallel machines with lookahead information. J Sched, to appear. doi:10.1007/s10951-010-0192-y

McNaughton R (1959) Scheduling with deadlines and loss functions. Manag Sci 6:1–12
Muratore G, Schwarz UM, Woeginger GJ (2010) Parallel machine scheduling with nested job assignment restrictions. Oper Res Lett 38(1):47–50
Noga J, Seiden SS (2001) An optimal online algorithm for scheduling two machines with release times. Theor Comput Sci 268:133–143
Ou J, Leung JY-T, Li C-L (2008) Scheduling parallel machines with inclusive processing set restriction. Nav Res Logist 55(4):328–338
Park J, Chang SY, Lee K (2006) Online and semi-online scheduling of two machines under a grade of service provision. Oper Res Lett 34:692–696
Pruhs K, Sgall J, Torng E (2004) Online scheduling. In: Leung JY-T (ed) Handbook of scheduling: algorithms, models, and performance analysis. CRC Press, Boca Raton
Sgall J (1998) On-line scheduling. Lecture Notes in Computer Science 1442:196–231
Shchepin EV, Vakhania N (2005) An optimal rounding gives a better approximation for scheduling unrelated machines. Oper Res Lett 33:127–133
Shmoys DB, Wein J, Williamson DP (1995) Scheduling parallel machines on-line. SIAM J Comput 24(6):1313–1331
Tan Z, Zhang A (2010a) A note on hierarchical scheduling on two uniform machines. J Comb Optim 20(1):85–95
Tan Z, Zhang A (2010b) Online hierarchical scheduling: an approach using mathematical programming. Theorl Comput Sci. doi:10.1016/j.tcs.2009.08.014
Wang Z, Xing W (2010) Worst-case analysis for on-line service polices. J Comb Optim 19:107–122
Wang Z, Xing W, Chen B (2009) On-line service scheduling. J Sched 12(1):31–43
Wu Y, Yang Q (2010) Optimal semi-online scheduling algorithms on two parallel identical mamchhines under a Grade of Service provision. Lecture Notes in Computer Science 6124:261–270
Zhang A, Jiang Y, Tan Z (2009) Online parallel machines scheduling with two hierarchies. Theor Comput Sci 410:3597–3605