Online Makespan Scheduling of Linear Deteriorating Jobs on Parallel Machines*
(Extended Abstract)

Sheng Yu[1], Jude-Thaddeus Ojiaku[2], Prudence W. H. Wong[2], and Yinfeng Xu[3]

[1] School of Business Administration, Zhongnan University of Economics and Law, Wuhan, China. yusheng_znufe@foxmail.com
[2] Department of Computer Science, University of Liverpool, UK
{J.Ojiaku, pwong}@liverpool.ac.uk
[3] School of Management, Xi'an Jiaotong University, China
yfxu@mail.xjtu.edu.cn

Abstract. Traditional scheduling assumes that the processing time of a job is fixed. Yet there are numerous situations that the processing time increases (deteriorates) as the start time increases. Examples include scheduling cleaning or maintenance, fire fighting, steel production and financial management. Scheduling of deteriorating jobs was first introduced on a single machine by Browne and Yechiali, and Gupta and Gupta independently. In particular, lots of work has been devoted to jobs with linear deterioration. The processing time p_j of job J_j is a linear function of its start time s_j, precisely, $p_j = a_j + b_j s_j$, where a_j is the normal or basic processing time and b_j is the deteriorating rate. The objective is to minimize the makespan of the schedule.

We first consider simple linear deterioration, i.e., $p_j = b_j s_j$. It has been shown that on m parallel machines, in the online-list model, LS (List Scheduling) is $(1 + b_{\max})^{1-\frac{1}{m}}$-competitive. We extend the study to the online-time model where each job is associated with a release time. We show that for two machines, no deterministic online algorithm is better than $(1 + b_{\max})$-competitive, implying that the problem is more difficult in the online-time model than in the online-list model. We also show that LS is $(1+b_{\max})^{2(1-\frac{1}{m})}$-competitive, meaning that it is optimal when $m = 2$.

1 Introduction

Makespan scheduling of deteriorating jobs. Scheduling jobs (with fixed processing time) on single or parallel machines is a classical problem [24]. Yet, there are numerous situations that the processing time increases (deteriorates) as the start time increases. For example, to schedule maintenance or cleaning, a delay often requires additional efforts to accomplish the task. Other examples are found in fire fighting, steel production and financial management [15, 21]. Scheduling of deteriorating jobs was first introduced by Browne and Yechiali [3], and Gupta

* This work is partially supported by NSF of China under Grants 71071123, 60736027 and 71101106. The work is partly done while Sheng Yu was at Xi'an Jiaotong University and was visiting University of Liverpool.

and Gupta [11] independently. Both considered minimizing makespan on a single machine. In [3], the processing time of a job is a monotone linear function of its start time while non-linear functions are considered in [11]. Since then, the problem has attracted a lot of attention, and has been studied in other time dependent models. Comprehensive surveys can be found in [1, 6, 9], which also discussed other objective functions.

Linear deterioration. We focus on jobs with *linear deterioration*, which has been studied in more detail due to its simplicity while capturing the essence of real life situations. The processing time of a job is a monotone linear function of its start time. Precisely, the processing time p_j of a job J_j is expressed as $p_j = a_j + b_j s_j$, where $a_j \geq 0$ is the "normal" or "basic" processing time, $b_j > 0$ is the deteriorating rate, and s_j is the start time. As the start time gets larger, the actual processing time also gets larger.

Linear deterioration is further said to be *simple* if $a_j = 0$, i.e., $p_j = b_j s_j$. In this case, in order to avoid trivial solution, it is natural to assume that the start time of the first job is $t_0 > 0$ since a start time of zero means that the processing time of all jobs is zero. Mosheiov [20, 21] justified simple linear deterioration as follows: as the number of jobs increases, the start time of jobs gets larger, and the actual processing time of infinitely many jobs is no longer affected by the normal processing time but only by the deteriorating rate.

Single and parallel machine scheduling. Non-preemptive scheduling of jobs with linear deterioration has been studied in both single and parallel machines settings.[4] The study first focuses on a single machine and all jobs are assumed to be available for processing at the same time. Gupta and Gupta [11] observed that with linear deterioration, it is optimal to process jobs in ascending order of a_j/b_j. With simple linear deteriorating rate, the makespan is indeed independent of the order of processing [21][5]. On parallel machines, the problem becomes intractable; it is NP-hard for two machines and strongly NP-hard for m machines because of the complexity of the corresponding problems with fixed processing time [7]. Kang and Ng [13] proposed an FPTAS. For simple linear deterioration, Kononov [14] and Mosheiov [22] independently showed that the problem is NP-hard, and Ren and Kang [26] proposed an FPTAS.

Release times and online algorithms. The above results assume that all jobs are available at the same time and full job information (a_j and b_j) is known in advance. In practice, jobs may be released at arbitrary times. We may also have to make decisions based on the jobs currently presented without information of future jobs. Pruhs, Sgall and Torng [25] formalized two online models. In the *online-list* model, jobs are available to be processed at the beginning but are presented one by one. Each job is to be allocated to a machine and a time period to execute before the next job is presented. Such allocation cannot be changed once it is made. In the *online-time* model, jobs are released at arbitrary times and a job is only known at its release time. The performance of online algorithms is typically measured by competitive analysis [2]. An online algorithm is c-competitive if for any input instance, its cost is no more than c times that of the optimal offline algorithm.

[4] Preemptive scheduling of linear deteriorating jobs has been studied on single machine [23].

[5] The makespan of running jobs $J_1, \cdots J_n$ equals $t_0(1 + b_1) \cdots (1 + b_n)$.

For online parallel machine scheduling with fixed processing time, Graham [10] has proposed the List Scheduling (LS) algorithm that schedules each job in turn to the machine that can complete the job the earliest. He showed that LS is $(2 - \frac{1}{m})$-competitive. Online algorithms and jobs with release times have been studied extensively for fixed processing time [25]. Yet, not much is known for deteriorating jobs with release times, let alone online algorithms.

For linear deteriorating jobs, jobs with release times have been studied in [5, 19]. In particular, it is explained in [19] in the application of steel production how jobs with release time and deteriorating rate apply in the scenario. Cheng and Ding [5] studied the complexity of the problem with release times, showing that on a single machine it is strongly NP-hard for identical normal processing time a or identical deteriorating rate b. Lee, Wu and Chung [19] proposed some heuristics for the case of identical deteriorating rate and evaluated them by experiments. The only work on online scheduling of deteriorating jobs that we are aware of is by Cheng and Sun [4]. They considered parallel machine scheduling of jobs with simple linear deterioration and the online-list model, showing that LS is $(1+b_{\max})^{1-\frac{1}{m}}$-competitive, where $b_{\max}$ is the maximum deteriorating rate. As we will show later, this is the best possible for any deterministic online-list algorithms. Several questions arise immediately following the work of [4].

- Intuitively, the problem becomes more difficult with arbitrary release times. In particular, for simple linear deterioration, can we show a lower bound larger than $(1 + b_{\max})^{1-\frac{1}{m}}$? Furthermore, what is the performance of LS for simple linear deterioration when jobs have arbitrary release times?
- What is the performance of LS or other online algorithms for other linear deterioration functions like $p_j = a_j + bs_j$, $p_j = a + b_j s_j$, and $p_j = a_j + b_j s_j$.

Availability constraints. We further consider the scenario when a machine is not always available [16–18]. This may arise due to maintenance or when the machine is reserved for other purposes. When a job is interrupted by an unavailable period, it may be resumed later or it may not be resumed and has to restart again. We focus on *non-resumable availability constraint*.

This problem has been studied in single machine scheduling of simple linear deteriorating jobs. Gawiejnowicz [8] proved that the problem is NP-hard for one unavailable period and strongly NP-hard for an arbitrary number of unavailable periods, while an FPTAS has been proposed for one unavailable period [12]. As far as we know, the only work on online algorithms shows that LS is an optimal online-list algorithm for one unavailable period [12], with a competitive ratio B/t_0 where $B > t_0$ is the beginning time of the unavailable period. An immediate question is to extend the study to parallel machines.

Our contribution. In this paper, we take one step forward to answer some of the above questions. The main results are for simple linear deterioration, where we first consider jobs with arbitrary release times. When scheduling on parallel machines, we show that the problem with arbitrary times is more difficult: for two machines, we give a lower bound of $1 + b_{\max}$ on the competitive ratio, which is strictly larger than the bound $(1 + b_{\max})^{1/2}$ for the online-list model. We also show that the competitive ratio of LS is between $1+b_{\max}$ and $(1+b_{\max})^{2(1-\frac{1}{m})}$ for m machines, while the ratio of RR (Round Robin) can be unbounded. Together with the general lower bound, it implies that for two machines, LS is optimal with competitive ratio $1 + b_{\max}$.

We then consider scheduling on two machines where one of them is unavailable during the interval $[B, F]$, with $t_0 < B < F$. For the online-list model, we show that a modified LS algorithm is optimal with a competitive ratio of $\min\{\sqrt{B/t_0}, 1 + b_{\max}\}$.

Another linear deterioration function we consider is with identical deteriorating rate but different normal processing times, i.e., $p_j = a_j + bs_j$. In this case, we focus on the online-list model. Let $a_{\max}$ and $a_{\min}$ be the maximum and minimum a_j, respectively, and $\alpha = \frac{a_{\max}}{a_{\min}}$. We show that no online-list algorithm is better than α-competitive, and show that Round Robin (RR) achieves this competitive ratio. We also show that LS is α-competitive in a special case.

Technically speaking, the lower bounds for simple linear deterioration are more technically involved. The adversaries work in stages. In each stage some jobs with small b are released, forcing the online algorithm to schedule jobs evenly on all machines; this is followed by a job with large b, forcing a large completion time. On the other hand, the optimal algorithm reserves a machine for the job with large b and then evenly distribute the remaining jobs on the other machines to achieve the same makespan for each machine. The idea is similar to those for fixed processing time, yet the crux is to work out the values of b's. To analyze the performance of LS, the key idea is to give a lower bound on the makespan of the optimal algorithm in terms of the completion time of the machine having the makespan in LS.

Organization of the paper. Section 2 gives some notations and definitions. In Section 3, we consider varying deteriorating rates while in Section 4, we consider varying normal processing times. Finally, we conclude in Section 5.

2 Preliminaries

We are to schedule a set of n jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ onto m machines $M_1, M_2, \ldots, M_m$. For every job J_j, we denote by r_j and p_j the *release time* and *processing time*, respectively. We assume that the jobs are indexed in increasing order of release time, i.e., $r_j \leq r_{j+1}$. The processing time p_j depends on the *start time* s_j when the job starts being executed by a processor, i.e., p_j differs with different schedules. In particular, we consider *linear deterioration* in which jobs are characterized by a normal processing time $a_j \geq 0$ and a deteriorating rate $b_j > 0$ such that $p_j = a_j + b_j s_j$. When the normal processing time is identical, it is called *simple linear deterioration*. Denote by $b_{\max}$ the maximum of b_j, $a_{\max}$ and $a_{\min}$ the maximum and minimum of a_j, and $\alpha = a_{\max}/a_{\min}$.

Jobs arrive online and the information about the jobs are only known on arrival. A schedule is to dispatch the jobs in $\mathcal{J}$ on machines and determine when to run the jobs. Preemption is not allowed. Consider a schedule S. For $1 \leq j \leq n$, the completion time of job J_j in S is denoted by $c_j(S)$. For any $1 \leq k \leq m$, the set of jobs dispatched to M_k by S is denoted by $\mathcal{J}^{(k)}(S)$. We simply use $\mathcal{J}^{(k)}$ when the context is clear. The makespan of machine M_k, denoted by $C^{(k)}_{\max}(S)$, is the maximum completion time of the jobs on M_k by S. The makespan of S, denoted by $C_{\max}(S)$, is the maximum of the makespan over all machines. I.e., $C^{(k)}_{\max}(S) = \max_{j \in \mathcal{J}^{(k)}(S)}\{c_j(S)\}$ and $C_{\max}(S) = \max_{1 \leq k \leq m}\{C^{(k)}_{\max}(S)\}$. The objective of the problem is to minimize the makespan of the schedule produced.

We also consider the case when there is an availability constraint. In this case, we consider only two machines and assume that machine M_1 has a known unavailable period $[B, F]$, where $t_0 < B < F$. The *semi-online* algorithm we propose requires $b_{\max}$ is known in advance.

Round Robin (RR). Jobs are inserted into a list in increasing order of arrival. The first job is dispatched to the first machine, and the next job is dispatched to the next machine, i.e., J_j is dispatched to M_k where $k = ((j-1) \bmod m)+1$.

List Scheduling (LS). Jobs are inserted into a list in increasing order of arrival. Whenever a machine becomes idle, the next job in the job list is dispatched to the machine.

OPT. We denote the optimal offline algorithm (and its schedule) by OPT.

3 Simple linear deterioration $p_j = b_j s_j$

In this section, we consider jobs with simple linear deteriorating rate, $p_j = b_j s_j$, i.e., $a_j = 0$. As we assume the normal processing time $a_j = 0$ for all j, it is natural to assume that the start time of the schedule is $t_0 > 0$ instead of 0, otherwise, p_j would all be zero. We consider two scenarios. In Section 3.1, we consider scheduling jobs with arbitrary release times on m machines. In Section 3.2, we consider the case when one of the machines may be unavailable for a certain period of time. In particular, we consider the special case with two machines and all jobs are available at t_0.

3.1 Online-time model: Scheduling jobs with arbitrary release times

In this section, we consider jobs with arbitrary release time r_j. We first make some simple observations (proof in full paper).

Property 1. Consider scheduling of jobs with $p_j = b_j s_j$.

(i) The completion time of any job J_j, $c_j = s_j(1 + b_j)$.

(ii) Consider any job set $\mathcal{J}$ where r denotes the earliest release time of jobs in $\mathcal{J}$. The makespan of scheduling these jobs on a single machine is at least $r\prod_{j\in\mathcal{J}}(1 + b_j)$.

(iii) Suppose $J_1, J_2, \cdots, J_n$ are indexed in increasing order of release times such that $r_j \le r_{j+1}$. The makespan of any m-machine schedule is at least

$$\prod_{1\le k\le m} r_k^{\frac{1}{m}} \prod_{1\le j\le n} (1 + b_j)^{\frac{1}{m}} \ .$$

Lower bounds. We prove several lower bounds showing that RR is not competitive (proof in full paper), and that the problem with arbitrary release times admits a larger competitive ratio than that for the online-list model.

Lemma 1. *Consider simple linear deteriorating jobs. The competitive ratio of* RR *is unbounded. This also holds for the online-list model.*

Lemma 2. *Consider simple linear deteriorating jobs with arbitrary release times. The competitive ratio of* LS *is at least* $(1 + b_{\max})$.

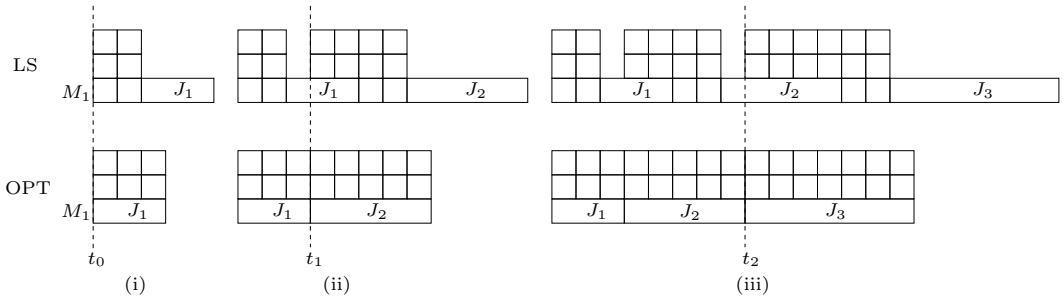

Fig. 1. The first three stages of the adversary for LS on three machines. The deteriorating rates b_1, b_2 and b_3 of J_1, J_2 and J_3 satisfy $1 + b_1 = (1 + b)^3$, $1 + b_2 = (1 + b)^5$, and $1 + b_3 = (1 + b)^7$. Note that the machines in OPT are busy all the time while those in LS may be idle. (i) In Stage 1, jobs are released at t_0. (ii) In Stage 2, newly arriving jobs have a release time of $t_1 = t_0(1 + b_1)$ and cannot be scheduled earlier on M_2 and M_3 in LS. (iii) In Stage 3, the release time $t_2 = t_1(1 + b_2)$. (Note that the length of the jobs in the figure reflects the value of deteriorating rates but not the actual processing time, which is increasing as start time increases.)

Proof (Sketch). The adversary works in stages and jobs are released at time t_i in Stage i, with $t_0 > 0$. In each stage, the adversary releases some jobs with deteriorating rate b at time t_i. LS would schedule these jobs evenly on the machines. Then one job of a large deteriorating rate b_i is released at t_i and no matter which machine LS schedules this job the completion time is big. On the other hand, the optimal offline algorithm OPT can reserve one machine for the job with large deteriorating rate and schedule the jobs with small deteriorating rate on the remaining machines. This introduces a difference in the latest completion time between LS and OPT in the current stage. The idea is similar to traditional scheduling with fixed processing time and the main issue is to choose appropriate b_i. Figure 1 shows the first three stages of the adversary. As to be shown in the full paper, we define a sequence b_i such that the difference between LS and OPT keeps increasing, and finally leading to the competitive ratio stated. □

The first stage of the above adversary can be used to show a lower bound on any deterministic algorithm for the online-list model (proof in full paper).

Lemma 3. *Consider simple linear deteriorating jobs. No deterministic online algorithm is better than* $(1+b_{\max})^{1-\frac{1}{m}}$*-competitive. This also holds for the online-list model.*

Next we consider two machines and extend the adversary in Lemma 2 to show that when jobs have arbitrary release times, no deterministic online algorithm is better than $(1 + b_{\max})$-competitive. The main difference is that after releasing jobs of deteriorating rate b, the online algorithm not necessarily schedules these jobs evenly between the two machines. Therefore, before the adversary releases a job of deteriorating rate b_k, more jobs are released to maintain the same completion time on both machines and the number and deteriorating rate of these intermediate jobs vary according to how the online algorithm schedule the set of jobs of deteriorating rate b.

Theorem 1. *Consider two-machine scheduling of jobs with arbitrary release times and simple linear deteriorating rates. No deterministic online algorithm is better than* $(1 + b_{\max})$*-competitive.*

Upper bound for LS. We prove that LS is at most $(1 + b_{\max})^{2(1-\frac{1}{m})}$-competitive when jobs have arbitrary release times. First of all, we claim that without loss of generality, we can assume that at any time in the LS schedule, not all machines are idle. Otherwise, suppose t is the latest time such that all machines are idle. As LS is not idle whenever there are available jobs, this means that there is a subset of jobs $\mathcal{J}' \subseteq \mathcal{J}$, all of which have release times strictly after t, and $\mathcal{J} - \mathcal{J}'$ are completed by LS before t. The makespan of LS on $\mathcal{J}'$ would remain the same as on $\mathcal{J}$. On the other hand, the makespan of OPT on $\mathcal{J}'$ is also the same as on $\mathcal{J}$ since there is a possible schedule to complete all the jobs in $\mathcal{J} - \mathcal{J}'$ before t. Therefore, we can make the following assumption without affecting the competitive ratio of LS.

Observation 2 *Without loss of generality, we may assume that at any time in a* LS *schedule, not all machines are idle.*

Let ℓ be the index of a job with completion time $C_{\max}(\text{LS})$; break ties arbitrarily. Let M_p be the machine to which LS schedules J_ℓ. Because of the way LS schedules job, we have the following property about $s_\ell(\text{LS})$, the start time of J_ℓ.

Property 2. (i) $s_\ell(\text{LS}) = C^p_{\max}(\text{LS})/(1+b_\ell)$. (ii) $s_\ell(\text{LS}) \leq \min_{1\leq k\leq m, k\neq p}\{C^{(k)}_{\max}(\text{LS})\}$.

To analyze the performance of LS, we extend the analysis in [4] first to input such that there is no idle time in the LS schedule, showing that LS is $(1 + b_{\max})^{1-\frac{1}{m}}$-competitive. In the general case when LS schedule may contain idle time, we use Lemma 2 to upper bound the makespan of LS and use an averaging argument to lower bound the makespan of OPT. The following theorem states the competitive ratio of LS (proof in full paper).

Theorem 3. *Consider m-machine scheduling of jobs with arbitrary release times and simple linear deteriorating rate,* LS *is $(1 + b_{\max})^{2(1-\frac{1}{m})}$-competitive.*

Corollary 1. *For two machine scheduling of jobs with arbitrary release times and simple linear deteriorating rate,* LS *is an optimal online algorithm with competitive ratio $1 + b_{\max}$.*

3.2 Online-list model: Two machine scheduling with availability constraint

In this section, we schedule jobs on two machines M_1 and M_2, where M_1 is unavailable during the period $[B, F]$. In particular, we consider *non-resumable availability constraint*, i.e., if a job is partly processed on M_1 before B, it has to be restarted from the beginning when M_1 becomes available at F. We consider jobs that are available at t_0, and have simple linear deteriorating rate. The online algorithm has to schedule a job as it is given, before the next job is presented, and the decision cannot be revoked. Furthermore, we assume that the online algorithm knows the maximum deteriorating rate $b_{\max}$ in advance, so we are considering *semi-online* algorithm. We first give a lower bound of $\min\{\sqrt{B/t_0}, 1 + b_{\max}\}$ on the competitive ratio. Then, we give a modified LS algorithm and show that it is $\min\{\sqrt{B/t_0}, 1 + b_{\max}\}$-competitive, implying that the algorithm is optimal.

It has been observed that on a machine that is entirely available, scheduling jobs with whatever order gives the same makespan. However, this is not the case if the machine is unavailable at some time. For example, given two jobs J_1 and J_2 with deteriorating rate b_1 and b_2 such that $1+b_1 = 1+\epsilon$ and $1+b_2 = B/t_0$. Processing J_1 before J_2 leads to a makespan of $F(1+b_2) = FB/t_0$ since J_2 cannot be completed before B and has to be started at F. Processing J_2 before J_1 leads to a makespan of $F(1+\epsilon)$.

Property 3. On the machine with unavailability, scheduling jobs in different order may result in different makespan.

Lower bounds. First, when $t_0(1+b_{\max}) \leq B$, we show in Lemma 4 that no deterministic online-list algorithm is better than $\min\{\sqrt{B/t_0}, 1+b_{\max}\}$-competitive. We present two adversaries, one for $\sqrt{B/t_0} < 1+b_{\max}$ and one for $\sqrt{B/t_0} \geq 1+b_{\max}$. Second, when $t_0(1+b_{\max}) > B$, we show in Lemma 5 (proof in full paper) that no deterministic online algorithm is better than $(1+b_{\max})$-competitive. Then in Section 3.2, we give matching upper bounds.

Lemma 4. *Suppose one of the two machines is unavailable during $[B, F]$ and jobs have simple linear deteriorating rates. When $t_0(1+b_{\max}) \leq B$, no deterministic online-list algorithm is better than $\min\{\sqrt{B/t_0}, 1+b_{\max}\}$-competitive.*

Proof (Sketch). Consider any online algorithm $\mathcal{A}$. We show here an adversary for the case $\sqrt{B/t_0} \geq 1+b_{\max}$. The case $\sqrt{B/t_0} < 1+b_{\max}$ is given in the full paper. The adversary first gives six jobs of the same deteriorating rate b_1 such that $1+b_1 = (B/t_0)^{1/4}$. If $\mathcal{A}$ does not schedule three jobs on each machine, then the adversary stops. Otherwise, the adversary then gives two jobs of deteriorating rate b_2 such that $1+b_2 = (1+\epsilon)(B/t_0)^{1/4}$, for some small ϵ. In both cases, one can show that the ratio can be made arbitrarily close to $1+b_{\max}$. □

Lemma 5. *Suppose one of the two machines is unavailable during $[B, F]$ and jobs have simple linear deteriorating rates. When $t_0(1+b_{\max}) > B$, no deterministic online-list algorithm is better than $(1+b_{\max})$-competitive.*

Upper bound. We modify the LS algorithm to cater for the unavailability period on M_1 and we call the algorithm MLS. MLS distinguishes the cases of B being small and large: $t_0(1+b_{\max}) \leq B$ and $t_0(1+b_{\max}) > B$.

Modified LS (MLS). If $t_0(1+b_{\max}) \leq B$, the interval $[t_0, t_0(1+b_{\max})]$ on M_1 is reserved to process the first job with deteriorating rate $b_{\max}$; otherwise, no interval is reserved. Apart from the job for which a time interval is reserved, a given job is scheduled on the machine that results in the minimum completion time. This includes three options: scheduling on M_1 before B (if the job can be completed before B), scheduling on M_1 after F, and scheduling on M_2.

Let J_ℓ be the job with the maximum completion time by MLS and M_p be the machine MLS schedules J_ℓ. Note that Property 2 (i) remains valid for MLS but (ii) only holds under the condition that $s_\ell > F$.

Property 4. (i) $s_\ell(\text{MLS}) = C^{(p)}_{\max}(\text{MLS})/(1+b_\ell)$. (ii) If $s_\ell(\text{MLS}) > F$, then $s_\ell(\text{MLS}) \leq C^{(k)}_{\max}(\text{MLS})$, where $k \neq p$.

Furthermore, we give a lower bound on $C_{\max}(\text{OPT})$ when $s_\ell \geq B$ (Property 5, proof in full paper) Let C_B be the latest completion time of the MLS schedule on M_1 before B. This property is used in Lemmas 6 and 7.

Property 5. If $s_\ell(\text{MLS}) \geq B$, then $C_{\max}(\text{OPT}) \geq s_\ell(\text{MLS})\sqrt{(1+b_\ell)\frac{C_B}{B}} > s_\ell(\text{MLS})$.

We then proceed to analyze the performance of MLS, showing that the upper bounds match the lower bounds in Lemmas 4 and 5.

Lemma 6. *Suppose one of the two machines is unavailable during $[B, F]$ and jobs have simple linear deteriorating rates. When $t_0(1+b_{\max}) \leq B$,* MLS *is a semi-online-list algorithm and is $\min\{\sqrt{B/t_0}, 1+b_{\max}\}$-competitive.*

Proof. Given any job set $\mathcal{J}$, we consider the case $s_\ell \geq B$ here and give the proof of the case $s_\ell < B$ in the full paper

According to how MLS schedules jobs, J_ℓ cannot be scheduled on M_1 at C_B, implying that $C_B(1+b_\ell) > B$. Furthermore, since we are considering the case $t_0(1+b_{\max}) \leq B$, it means that MLS reserves the interval $[t_0, t_0(1+b_{\max})]$ for the first job with $b_{\max}$, and hence, $t_0(1+b_{\max}) \leq C_B$. We can then obtain two bounds on the ratio $C_{\max}(\text{MLS})/C_{\max}(\text{OPT})$. First, by Property 5, $C_{\max}(\text{OPT}) \geq s_\ell$.

$$\frac{C_{\max}(\text{MLS})}{C_{\max}(\text{OPT})} \leq \frac{s_\ell(1+b_\ell)}{s_\ell} \leq 1+b_{\max} \ .$$

Furthermore, by Property 5, $C_{\max}(\text{OPT}) \geq s_\ell\sqrt{(1+b_\ell)\frac{C_B}{B}}$.

$$\frac{C_{\max}(\text{MLS})}{C_{\max}(\text{OPT})} \leq \frac{s_\ell(1+b_\ell)}{s_\ell\sqrt{(1+b_\ell)\frac{C_B}{B}}} = \sqrt{1+b_\ell}\sqrt{B/C_B} \leq \sqrt{1+b_{\max}}\sqrt{B/C_B} \leq \sqrt{B/t_0} \ ,$$

where the last inequality is a consequence of $t_0(1+b_{\max}) \leq C_B$. Therefore, the ratio is at most $\min\{\sqrt{B/t_0}, 1+b_{\max}\}$ as required. □

The proof of Lemma 7 is given in the full paper.

Lemma 7. *Suppose one of the two machines is unavailable during $[B, F]$ and jobs have simple linear deteriorating rates. When $t_0(1+b_{\max}) > B$,* MLS *is a semi-online-list algorithm and is $(1+b_{\max})$-competitive.*

By Lemmas 4, 5, 6, and 7, we have the following corollary.

Corollary 2. *Consider two machines one of which is unavailable during $[B, F]$ and jobs with simple linear deteriorating rates.* MLS *is an optimal semi-online-list algorithm.*

4 Online-list model: Fixed deteriorating rate and varying normal processing time $p_j = a_j + b\ s_j$

In this section, we consider jobs with fixed deteriorating rate but varying normal processing time. We focus on the online-list model in which jobs are presented one by one. When a job is given, it is available for process, then the online

algorithm has to dispatch the job to a machine and specify the period of time to process the job. This decision has to be made before the next job is given, and cannot be changed once it is made. The machines are available for processing starting from time 0.

Recall that $\mathcal{J}^{(k)}(S)$ denote the set of jobs dispatched on machine M_k by schedule S and $n^{(k)}(S)$ is the size of $\mathcal{J}^{(k)}(S)$. Suppose $\mathcal{J}^{(k)}(S) = \{J_{k,1}, J_{k,2}, \cdots, J_{k,n^{(k)}}\}$. The completion time of $J_{k,1}$ denoted by $c_{k,1}$ equals to $a_{k,1}$. For the second job, $c_{k,2} = a_{k,2} + a_{k,1}(1+b)$. In general, for any $1 \leq j \leq n^{(k)}$, the completion time of the j-th job is

$$c_{k,j} = \sum_{1 \leq i \leq j} a_{k,i}(1+b)^{j-i} \ .$$

Therefore, given a set of jobs on a particular machine, the optimal offline schedule is to schedule jobs in increasing order of normal processing time a_j. Recall that $a_{\max}$ and $a_{\min}$ denote the maximum and minimum value of a_j, and α denotes the ratio $\frac{a_{\max}}{a_{\min}}$.

4.1 Lower bounds

In this section, we give a lower bound on any online-list algorithm.

Theorem 4. *Consider the online-list model with jobs having fixed deteriorating rate. On m machines, the competitive ratio of any online-list algorithm is no better than α.*

Proof (Sketch). We present the adversary and leaves the detail analysis in the full paper. Consider any online-list algorithm $\mathcal{A}$. The adversary first releases mq jobs all with normal processing time a_1, for some positive integer q. If $\mathcal{A}$ schedules $q+1$ or more jobs on one of the machines, the adversary stops releasing jobs. Otherwise, q jobs are dispatched on each machine. The adversary releases another mq jobs with normal processing time $a_2 < a_1$. In both cases, we can show that $C_{\max}(\mathcal{A})/C_{\max}(\text{OPT})$ can be made arbitrarily close to α. □

4.2 Upper bounds

In this section, we derive upper bounds on the performance of online algorithms (proofs in full paper). First of all, we observe that if there is only one machine, LS is at most α-competitive. We can then extend this proof to show that RR is α-competitive for parallel machines.

Lemma 8. *Consider the online-list model with jobs having fixed deteriorating rate. On a single machine, the competitive ratio of* LS *is at most α.*

When we consider parallel machines, we notice that for any schedule, the machine with the maximum number of jobs has at least $\lceil \frac{n}{m} \rceil$ jobs. On the other hand, the algorithm RR schedules at most $\lceil \frac{n}{m} \rceil$ jobs to any machine. Using the same argument as Lemma 8, we have

$$\frac{C_{\max}(\text{RR})}{C_{\max}(\text{OPT})} \leq \frac{a_{\max} \sum_{1 \leq j \leq \lceil \frac{n}{m} \rceil} (1+b)^{\lceil \frac{n}{m} \rceil - j}}{a_{\min} \sum_{1 \leq j \leq \lceil \frac{n}{m} \rceil} (1+b)^{\lceil \frac{n}{m} \rceil - j}} = \alpha \ .$$

Then we have the following theorem.

Theorem 5. *Consider the online-list model with jobs having fixed deteriorating rate. On m machines, the competitive ratio of* RR *is at most α.*

Notice that the maximum number of jobs LS schedules to a machine may be more than $\lceil \frac{n}{m} \rceil$. Nevertheless, the next lemma asserts that under the condition that $\alpha \leq 1 + b$, a machine processing more jobs always has a larger makespan. In this case, LS schedules at most $\lceil \frac{n}{m} \rceil$ jobs to any machine, implying that LS is α-competitive (Corollary 3).

Lemma 9. *Consider the online-list model with jobs having fixed deteriorating rate. On m machines, when $\alpha \leq 1 + b$, the makespan of a machine processing more jobs is larger than that of a machine processing fewer jobs.*

Corollary 3. *Consider the online-list model with jobs having fixed deteriorating rate. On m machines, when $\alpha \leq 1 + b$, the competitive ratio of* LS *is at most α.*

5 Summary and future work

In this paper, we study online parallel machine scheduling of jobs with linear deteriorating rate. Two linear deterioration functions have been considered. For $p_j = b_j s_j$ and jobs with release times, we show that LS is $(1 + b_{\max})^{2(1-\frac{1}{m})}$-competitive, where $b_{\max}$ is the maximum deteriorating rate. We also show that on m machines, no online algorithm is better than $(1 + b_{\max})^{1-\frac{1}{m}}$-competitive; and on two machines, no online algorithm is better than $(1 + b_{\max})$-competitive. We believe it is possible to extend the adversary for two machines to m machines. An obvious open question to close the gap between the upper and lower bounds.

As for the study of availability constraint, we have given an optimal online-list algorithm when there are two machines one of which has an unavailable period. Extensions include considering more than two machines, more than one unavailable periods, and/or more than one machines being unavailable. It is also interesting to extend the study to the online-time model where jobs have arbitrary release times.

For the linear deterioration function $p_j = a_j + bs_j$, we give a lower bound and show that RR achieves this competitive ratio. We believe LS also achieves this ratio but we manage to show it for a special case. An immediate question is to determine the competitive ratio of LS for all cases. Again it is interesting to extend the study to the online-time model.

Another direction is to consider more general functions like $p_j = a_j + b_j s_j$, non-linear deterioration, or other time dependent functions [9], e.g., decrease in processing time as start time increases captures the learning effect.

References

1. B. Alidaee and N. K. Womer. Scheduling with time dependent processing times: Review and extensions. *J. of Operational Research Society*, 50(7):711–720, 1999.
2. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, Cambridge, 1998.
3. S. Browne and U. Yechiali. Scheduling deteriorating jobs on a single processor. *Operations Research*, 38(3):495–498, 1990.

4. M. B. Cheng and S. J. Sun. A heuristic MBLS algorithm for the two semi-online parallel machine scheduling problems with deterioration jobs. *Journal of Shanghai University*, 11(5):451–456, 2007.
5. T. C. E. Cheng and Q. Ding. The complexity of single machine scheduling with release times. *Information Processing Letters*, 65(2):75–79, 1998.
6. T. C. E. Cheng, Q. Ding, and B. M. T. Lin. A concise survey of scheduling with time-dependent processing times. *European J. of OR*, 152(1):1–13, 2004.
7. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.
8. S. Gawiejnowicz. Scheduling deteriorating jobs subject to job or machine availability constraints. *European J. of OR*, 180(1):472–478, 2007.
9. S. Gawiejnowicz. *Time-Dependent Scheduling*. Springer-Verlag, Berlin, 2008.
10. R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.
11. J. N. D. Gupta and S. K. Gupta. Single facility scheduling with nonlinear processing times. *Computers and Industrial Engineering*, 14(4):387–393, 1988.
12. M. Ji, Y. He, and T. C. E. Cheng. Scheduling linear deteriorating jobs with an availability constraint on a single machine. *Theoretical Computer Science*, 362(1-3):115–126, 2006.
13. L. Y. Kang and C. T. Ng. A note on a fully polynomial-time approximation scheme for parallel-machine scheduling with deteriorating jobs. *International Journal of Production Economics*, 109(1-2):108–184, 2007.
14. A. Kononov. Scheduling problems with linear increasing processing times. In e. a. Zimmermann U, editor, *Operations Research Proceedings 1996,*. pages 208–212, Berlin, 1997.
15. A. S. Kunnathur and S. K. Gupta. Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem. *European Journal of Operation Research*, 47(1):56–64, 1990.
16. C. Y. Lee. Machine scheduling with an availability constraint. *Journal of Global Optimization*, 9(3-4):395–416, 1996.
17. C. Y. Lee. Machine scheduling with availability constraints. In J. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, pages 22.1–22.13. Chapman and Hall, Boca Raton, 2004.
18. C. Y. Lee, L. Lei, and M. Pinedo. Current trend in deterministic scheduling. *Annals of Operations Research*, 70:1–42, 1997.
19. W. Lee, C. Wu, and Y. Chung. Scheduling deteriorating jobs on a single machine with release times. *Computers and Industrial Engineering*, 54(3):441–452, 2008.
20. G. Mosheiov. V-shaped policies for scheduling deteriorating jobs. *Operations Research*, 39:979–991, November 1991.
21. G. Mosheiov. Scheduling jobs under simple linear deterioration. *Computers and Operations Research*, 21(6):653–659, 1994.
22. G. Mosheiov. Multi-machine scheduling with linear deterioration. *INFOR: Information Systems and Operational Research*, 36(4):205–214, 1998.
23. C. T. Ng, S. S. Li, T. C. E. Cheng, and J. J. Yuan. Preemptive scheduling with simple linear deterioration on a single machine. *Theoretical Computer Science*, 411(40-42):3578–3586, 2010.
24. M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice-Hall, Upper Saddle River, 2002.
25. K. Pruhs, J. Sgall, and E. Torng. Online scheduling. In J. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, pages 15.1–15.42. Chapman and Hall, Boca Raton, 2004.
26. C. R. Ren and L. Y. Kang. An approximation algorithm for parallel machine scheduling with simple linear deterioration. *Journal of Shanghai University*, 11(4):351–354, 2007.