Appl. Math. J. Chinese Univ.
2019, 34(1): 113-126

Online scheduling of jobs with kind release times and deadlines on a single machine

LI Wen-jie[1] MA Ran[2,*] FENG Qi[3]

Abstract. This paper studies online scheduling of jobs with kind release times on a single machine. Here "kind release time" means that in online setting, no jobs can be released when the machine is busy. Each job J has a kind release time $r(J) \geq 0$, a processing time $p(J) > 0$ and a deadline $d(J) > 0$. The goal is to determine a schedule which maximizes total processing time $(\sum p(J)E(J))$ or total number $(\sum E(J))$ of the accepted jobs. For the first objective function $\sum p(J)E(J)$, we first present a lower bound $\sqrt{2}$, and then provide an online algorithm LEJ with a competitive ratio of 3. This is the first deterministic algorithm for the problem with a constant competitive ratio. When $p(J) \in \{1, k\}$, $k > 1$ is a real number, we first present a lower bound $\min\{(1+k)/k, 2k/(1+k)\}$, and then we show that LEJ has a competitive ratio of $1 + \lceil k \rceil / k$. In particular, when all the k length jobs have tight deadlines, we first present a lower bound $\max\{4/(2+k), 1\}$ (for $\sum p(J)E(J)$) and $4/3$ (for $\sum E(J)$). Then we prove that LEJ is $\lceil k \rceil / k$-competitive for $\sum p(J)E(J)$ and we provide an online algorithm H with a competitive ratio of $2\lceil k \rceil / (\lceil k \rceil + 1)$ for the second objective function $\sum E(J)$.

§1 Introduction and Notation

Online scheduling problems have received extensive attention in the recent three decades. The state-of-the-art reviews on online scheduling were provided by Pruhs, et al. [10], Tan and Zhang [11], and Tian, et al. [13]. Three basic models have been proposed: online over list, online non-clairvoyance and online over time. In this paper, we study the online over time model. In this case, the characteristics of each job, including its release time, processing time and deadline, are unknown until it is released.

Received: 2016-12-02. Revised: 2018-01-01.
MR Subject Classification: 90B36, 90C39.
Keywords: scheduling, online algorithm, kind release time, deadline.
Digital Object Identifier(DOI): https://doi.org/10.1007/s11766-019-3512-9.
Supported by the National Natural Science Foundation of China (11501279, 11501171, 11671188, and 11401604) and the Young Backbone Teachers of Luoyang Normal University (2018XJGGJS-10) and Henan Colleges (2015GGJS-193).
*Corresponding author.

For a maximization problem, the competitive ratio $\rho_{\mathcal{A}}$ of an online algorithm $\mathcal{A}$ is defined to be $\rho_{\mathcal{A}} = \sup\{\mathrm{OPT}(I)/\mathcal{A}(I)\}$, where I is a sequence of jobs with $\mathrm{OPT}(I) > 0$. Here, for a sequence I of jobs, $\mathcal{A}(I)$ is the objective value of the schedule obtained by algorithm $\mathcal{A}$, and $\mathrm{OPT}(I)$ is the objective value of an off-line optimal schedule. An online algorithm $\mathcal{A}$ is called best possible if no online algorithm has a competitive ratio less than that of $\mathcal{A}$. In particular, when $\rho_{\mathcal{A}} = 1$, we say $\mathcal{A}$ is an optimal online algorithm for the problem. Occasionally, we use OPT to denote an optimal schedule.

Since the jobs' information is unknown in advance, most online algorithms have competitive ratios greater than 1. To improve the performance of online strategy, some beneficial scheduling environments are suggested in the literature, which include scheduling with preemption (Baker [2]), scheduling with restart (Hoogeveen, et al. [6], scheduling with lookahead (Keskinocak [7]), and so on. Online scheduling of jobs with kind release times is a new schedule environment which was firstly introduced by Li and Yuan [9]. "Kind release time (say KRT)" implies that in online setting, no jobs can be released when the machine is busy. The job instance generated in the KRT environment is dependent on the online strategy. This is different from the traditional online setting.

The research in this paper is motivated by the phenomenon that, the outpatient appointment in a hospital. Before going to the hospital, one patient (job) should first make an appointment (arrive time) with the doctor (machine). In general, the doctor can accept the appointment only when he or she is available. Moreover, KRT setting may help to improve the competitive ratio of online algorithm for some online problems. One typical example is the problem $1 \mid \text{online}, r_j \mid \sum w_j C_j$. Anderson and Potts [1] provided a 2-competitive best possible online algorithm for this problem. On the other hand, under the KRT setting, the online version of Smith's SWPT rule can solve the above problem optimally. Another typical example is the problem $Pm \mid \text{online}, r_j, \text{p-batch}, b = \infty \mid C_{\max}$. Here, "p-batch" means that the machine can process at most b jobs simultaneously as a batch. Jobs processed in the same batch have the same starting time and completion time; "$b = \infty$" implying that the batch capacity is unlimited. For this problem, the best possible online algorithm in Tian, et al. [12], and Liu, et al. [8] has a competitive ratio of $(2 - m + \sqrt{m^2 + 4})/2$. On the other hand, under the KRT setting, the online version of Graham's LS rule can solve the above problem optimally.

Suppose that, there are jobs to be scheduled on a single machine. The jobs arrive over time. Each job J has a release time $r(J) \geq 0$, a processing time $p(J) > 0$, and a deadline $d(J) > 0$. Especially, if $d(J) = r(J) + p(J)$, J is called a tight job, or equivalently, J has a tight deadline. For a given schedule, the completion time of job J is denoted by $C(J)$. J is called early (or on time) if $C(J) \leq d(J)$, and tardy otherwise. The early indicator number of job J is defined by $E(J) = 1$ if J is early, and $E(J) = 0$ if J is tardy. We assume that the tardy jobs will not be accepted by a schedule. So, if job J is accepted by a schedule, it must be an early job. The objective is to find a schedule so that the total processing time ($\sum p(J)E(J)$) or total number ($\sum E(J)$) of the accepted jobs is maximized. A job J is said to be effective at time t if $t + p(J) \leq d(J)$. We say a job J expires at time t if $t + p(J) > d(J)$.

Online scheduling of jobs with deadlines is a big research direction in the modern scheduling theory. So, there have been lots of results on this area. Here we only mention the results that are closely related to this paper. As the first positive result on the online version, Baruah, et al. [3] showed that a greedy algorithm is 2-competitive for $1 \mid \text{online}, r(J), p(J) = p \mid \sum E(J)$. In fact, they showed that 2-competitiveness holds for any non-preemptive deterministic algorithm that is never idle at times when some jobs are available for execution. Goldman, et al. [5] showed that the above greedy algorithm is a best possible online algorithm. When restart is allowed, Chrobak, et al. [4] provided a best possible online algorithm for $1 \mid \text{online}, r(J), p(J) = p \mid \sum E(J)$ which has a competitive ratio of 3/2. Here "restart" means that we may interrupt some running jobs and the processing of these jobs is wasted. Then interrupted jobs are released and become independently unscheduled jobs. Hoogeveen, et al. [6] provided a best possible online algorithm for $1 \mid \text{online}, r(J) \mid \sum E(J)$ which has a competitive ratio of 2. Goldman, et al. [5] first studied the problem $1 \mid \text{online}, r(J), p(J) \in \mathcal{P} \mid \sum p(J)E(J)$, where $\mathcal{P} = \{1, k\}, \{1, 2, \cdots, 2^c\}$, or $\{p(J) : 1 \le p(J) \le 2^c\}$, $k > 1$ is a real number and $c > 1$ is a positive integer. When $\mathcal{P} = \{1\}$ and $d(J) - r(J) \ge 2$, they provided a 3/2-competitive deterministic algorithm. For the case that $\mathcal{P} = \{1, k\}$, they obtained a 4-competitive randomized algorithm. If $\mathcal{P} = \{1, 2, ..., 2^c\}$ and $\mathcal{P} = \{p(J) : 1 \le p(J) \le 2^c\}$, they gave a randomized algorithm with a competitive ratio of $3(c+1)$ and $6(c+1)$, respectively. To the best of our knowledge, when the processing time of jobs is arbitrary, no results about the deterministic algorithm were reported in the literature.

§2 Problem formulation and main results

We study online scheduling of jobs with deadlines under the KRT setting on a single machine. The goal is to determine a schedule which maximizes $\sum p(J)E(J)$ and $\sum E(J)$. The introducing of the KRT setting is motivated by beating the 2-competitiveness of online algorithms in Hoogeveen, et al. [6]. Due to the hardness of this problem, we first study a special version. Suppose that the length of each job is either 1 or k (where $k > 1$ is a real number) and all the k length jobs have tight deadlines. On the other hand, we try to design a deterministic online algorithm for $1 \mid \text{online}, r(J) \mid \sum p(J)E(J)$ under the KRT environment. The problems studied in this paper can be denoted by $1 \mid \text{online}, \text{KRT}, r(J), p(J) \in \{1, k\} \mid \sum E(J)$ (or $\sum p(J)E(J)$) and $1 \mid \text{online}, \text{KRT}, r(J) \mid \sum p(J)E(J)$, respectively. We summarize the main results of this paper as follows:

• For $1 \mid \text{online}, \text{KRT}, r(J) \mid \sum p(J)E(J)$, we first present a lower bound $\sqrt{2}$, and then provide an online algorithm LEJ with a competitive ratio of 3.

• For $1 \mid \text{online}, \text{KRT}, r(J), p(J) \in \{1, k\} \mid \sum p(J)E(J)$, we first present a lower bound $\min\{(1 + k)/k, 2k/(1 + k)\}$, and then we prove that LEJ has a competitive ratio of $1 + \lceil k \rceil / k$.

• For $1 \mid \text{online}, \text{KRT}, r(J), p(J) \in \{1, k\} \mid \sum p(J)E(J)$, when all long jobs are tight, we first present a lower bound $\max\{4/(2 + k), 1\}$, and then we show that LEJ has a competitive ratio of $\lceil k \rceil / k$. This implies that LEJ is an optimal online algorithm for the case where k is a positive integer.

- For $1 \mid \text{online}, \text{KRT}, r(J), p(J) \in \{1, k\} \mid \sum E(J)$, when all long jobs are tight, we first present a lower bound 4/3, and then prove that H has a competitive ratio of $2\lceil k\rceil/(\lceil k\rceil + 1)$. This means that H is a best possible online algorithm for $1 < k \leq 2$.

§3 For the objective function: $\sum E(J)$

In this section, we study the problem $1 \mid \text{online}, \text{KRT}, r(J), p(J) \in \{1, k\} \mid \sum E(J)$ under the constraint that all the k length jobs are tight (i.e., $d(J) = r(J) + k$ for any long job J). We first provide a lower bound of 4/3 , then we design an online algorithm H and prove that the competitive ratio of H is $2\lceil k\rceil/(\lceil k\rceil + 1)$.

3.1 The lower bound

Theorem 3.1. *For problem $1 \mid$ online, KRT, $r(J), p(J) \in \{1, k\} \mid \sum E(J)$, no online algorithms have a competitive ratio less than* 4/3.

Proof. The jobs, written in the form $J_j = (r(J_j), p(J_j), d(J_j))$ are $J_1 = (0, k, k)$, $J_2 = (0, 1, d)$, $J_3 = (0, 1, d)$, $J_4 = (k, 1, d), J_5 = (k, 1, d)$, where $d = \max\{k, 2\} + 2$. Note that $4 \leq d < k + 3$. The instance I is the jobs set $\{J_1, J_2, J_3\}$ or $\{J_1, J_2, J_3, J_4, J_5\}$. For an arbitrary online algorithm $\mathcal{A}$, we distinguish the following two cases to construct the job instance I by the adversary.

Case 1. Algorithm $\mathcal{A}$ starts J_1 at time 0, then jobs J_4, J_5 arrive at time k. since $k > 1$ and $4 \leq d < k + 3$, there are at most two unit jobs scheduled during the time interval $[k, d]$ by $\mathcal{A}$. So, $\mathcal{A}(I) \leq 3$. Note that, OPT can schedule all the unit jobs in I. It follows that $\text{OPT}(I)/\mathcal{A}(I) \geq 4/3$.

Case 2. Algorithm $\mathcal{A}$ does not start J_1 at time 0, then no other jobs will arrive after time 0, and so $\mathcal{A}(I) \leq 2$ and $\text{OPT}(I) = 3$. Hence $\text{OPT}(I)/\mathcal{A}(I) \geq 3/2$.

Combining the above discussion, we have $\text{OPT}(I)/\mathcal{A}(I) \geq \min\{4/3, 3/2\} = 4/3$. □

3.2 Online algorithm H

We refer to the length 1 jobs as *short* jobs, and the length k jobs as *long* jobs. For convenience, a short job is indicated by J^a and a long job is indicated by J^b. Let t be the current time. A job J is said to be effective at time t if $t + p(J) \leq d(J)$. We say a job J expires at time t if $t + p(J) > d(J)$. We use $U(t)$ to denote the set of all unscheduled effective jobs (i.e., available but not expired at time t) at time t. Consider an arbitrary instance I of jobs. For each job $J^b \in I$, we have $d(J^b) = r(J^b) + k$. So, we can assume that at most one long job is released at one time point and $U(t)$ contains at most one long jobs for any time point t.

At time t, we call $U(t)$ a *complete job set* if EDD strategy can accept all jobs in $U(t)$. Note that, if $U(t)$ is a complete job set and contains a long job J^b, then J^b must be the job in $U(t)$ with the smallest deadline. Online algorithm H works as follows.

```
Algorithm H

Step 0: Set $t = 0$.

Step 1: If $U(t) = \emptyset$, then wait until a new job arrives and reset $t$ to be the release time of
    such job.

Step 2: If $U(t) \neq \emptyset$ and the machine is idle, then do the following.

    Step 2.1: If $U(t)$ contains no long jobs, then pick a short job from $U(t)$ with the smallest
        deadline and start it at time $t$. Reset $t := t + 1$, go to step 1. Otherwise, go to step
        2.2.

    Step 2.2: Assume that $U(t)$ contains a long job $J^b$. Then distinguish the following two
        cases.

    Step 2.2.1: If $U(t)$ is a complete job set, then start $J^b$ on the machine at time $t$. Reset
        $t := t + k$, go to step 1. Otherwise, go to step 2.2.2.

    Step 2.2.2: Obviously, $U(t)$ contains at least one short job. Then pick a short job from
        $U(t)$ with the smallest deadline and start it at time $t$. Reset $t := t + 1$, go to step 1.
```

3.3 Analysis

Consider any job instance I. Let $\sigma(I)$ denote the schedule generated by H for I. When no ambiguity can occur, we also use $\sigma(I)$ to denote the set of jobs accepted by H. We call a maximal subsequence $\sigma^b(I) \subseteq \sigma(I)$ a *big block*, if all jobs in $\sigma^b(I)$ are long jobs and these long jobs are processed continuously by H. Similarly, we call a maximal subsequence $\sigma^a(I) \subseteq \sigma(I)$ a *small block*, if all jobs in $\sigma^a(I)$ are short jobs and these short jobs are processed continuously by H. Let S^b, S^a, C^b, C^a be the starting time of the first job and the completion time of the last job in $\sigma^b(I)$ and $\sigma^a(I)$, respectively. For any job $J \in I$, we use $s(J)$, $s^*(J)$, $c(J)$, $c^*(J)$ to denote the starting time and completion time of J in H and OPT, respectively. For any other forms of big block (or small block) the above notation and terminology for $\sigma^b(I)$ (or $\sigma^a(I)$) are similarly understood.

Below, we will prove that the competitive ratio of algorithm H is $\rho = 2\lceil k\rceil/(\lceil k\rceil+1)$. Let I be a smallest counterexample about the problem (i.e., a counterexample consisting of a minimum number of jobs). Then $\text{OPT}(I) > \rho\text{H}(I)$. Recall that $\sigma(I)$ denote the schedule produced by H for the instance I. We use α and β to denote the number of short jobs and long jobs in $\sigma(I)$, respectively. Then we have $\text{H}(I) = |\sigma(I)| = \alpha + \beta$. We use C^0 and C^I to denote the release time of the first job and the completion time of the last job in $\sigma(I)$. Then we have the following two observations.

Observation 3.1. *Instance I must consist of short jobs and long jobs.*

Proof. If possible, assume that instance I only contains one kind jobs (short jobs or long jobs). Since all jobs in I have kind release times and algorithm H schedules jobs under EDD rule, H is an optimal algorithm for I. So, we have $\mathrm{H}(I) = \mathrm{OPT}(I)$. This contradicts the definition of I. □

Observation 3.2. *All jobs in $\sigma(I)$ are processed contiguously by H, i.e., there is no idle time interval between $[C^0, C^I)$.*

Proof. Suppose that there are idle time intervals between $[C^0, C^I)$. Let $[s, t] \subset [C^0, C^I)$ be the last such idle time interval. Set $V = \{J : r(J) \geq t\}$. Then we have $\mathrm{OPT}(I) \leq \mathrm{OPT}(I \setminus V) + \mathrm{OPT}(V)$. By H, $\mathrm{H}(I) = \mathrm{H}(I\setminus V)+\mathrm{H}(V)$. By the definition of I, we have $\mathrm{OPT}(I\setminus V) \leq \rho\mathrm{H}(I\setminus V)$ and $\mathrm{OPT}(V) \leq \rho\mathrm{H}(V)$. Consequently, $\mathrm{OPT}(I) \leq \rho\mathrm{H}(I)$. This contradicts the choice of I. □

By Observation 3.2, all jobs in $\sigma(I)$ are scheduled continuously. We divide the whole schedule $\sigma(I)$ into two sorts of blocks, big blocks $\sigma_j^b(I)$ and small blocks $\sigma_j^a(I)$ with $j = 0, 1, ..., m$. For each $0 \leq j \leq m$, we assume that $\sigma_j^b(I)$ and $\sigma_j^a(I)$ are adjacent to each other and $\sigma_j^b(I)$ is prior to $\sigma_j^a(I)$. Namely, the completion time of the last job in $\sigma_j^b(I)$ is just the starting time of the first job in $\sigma_j^a(I)$ (i.e., $C_j^b = S_j^a$). In particular, if $\sigma_0^b(I)$ (or $\sigma_m^a(I)$) does not exist, then we set $\sigma_0^b(I) = \emptyset$ (or $\sigma_m^a(I) = \emptyset$).

Observation 3.3. *For any short job $J^a \in I \setminus \sigma(I)$, $d(J^a) \notin \bigcup_{0\leq j\leq m}[S_j^b + 1, C_j^b + 1)$.*

Proof. For any short job $J^a \in I \setminus \sigma(I)$, we have $d(J^a) \in [C^0 + 1, C^I + 1)$. Suppose that there exists a short job $J^a \in I\setminus\sigma(I)$ and a long job $J^b \in \sigma_h^b(I)$ such that $d(J^a) \in [s(J^b)+1, c(J^b)+1) \subseteq [S_h^b + 1, C_h^b + 1)$, where $1 \leq h \leq m$. Since J^a has kind release time, J^a has been released at time $s(J^b)$. So, we have $J^a, J^b \in U(s(J^b))$. Note that, $U(s(J^b))$ is not a complete job set at time $s(J^b)$. By H, J^b cannot be scheduled at time $s(J^b)$. This contradicts the fact that $J^b \in \sigma_h^b(I)$. □

For each small block $\sigma_j^a(I) \subseteq \sigma(I)$, $j = 0, 1, ..., m$. Set $X_j = \sigma_j^a(I) \cup \{J^a \in I\setminus\sigma(I) : S_j^a + 1 \leq d(J^a) < C_j^a + 1\} \cup \{J^b \in I : r(J^b) \in [S_j^a, C_j^a)\}$. Note that, all long jobs from I are tight, if no long jobs arrive in the time period $[S_j^a, C_j^a)$. By the execution of algorithm H, we know that H schedules these short jobs in $\sigma_j^a(I)$ under EDD strategy. Moreover, since all jobs in I have kind release times, H is an optimal strategy for the job set X_j in $[S_j^a, \infty)$. Suppose that there exist some long jobs (say $J_1^b, J_2^b, ..., J_m^b$) in X_j. Since all jobs accepted by H are short jobs in $[S_j^a, C_j^a)$, we know that $U(t)$ is not a complete job set at any time $t \in \{r(J_1^b), r(J_2^b), ..., r(J_m^b)\}$. This means that if H schedules a long job at time t, then at least one short job will expire at time $t + k$. Note that, H still schedules these short jobs in $\sigma_j^a(I)$ under EDD strategy. Thus H is also an optimal strategy for X_j in $[S_j^a, \infty)$. By summing up the above analysis, we conclude that the number of jobs accepted by any algorithm for X_j in $[S_j^a, \infty)$ cannot be larger than $|\sigma_j^a(I)|$. For convenience, we say algorithm H is *locally optimal* for job set X_j during the time interval $[S_j^a, \infty)$. Then we have the following two lemmas.

Lemma 3.1. $\bigcup_{0\leq j\leq m} \sigma_j^a(I) \neq \emptyset$ *and* $\bigcup_{0\leq j\leq m} \sigma_j^b(I) \neq \emptyset$.

Proof. If $\bigcup_{0\le j\le m}\sigma_j^a(I)=\emptyset$, and by algorithm H, instance I only contains one sort of jobs (short jobs or long jobs). This contradicts Observation 3.1. If $\bigcup_{0\le j\le m}\sigma_j^b(I)=\emptyset$, then $\sigma(I)$ is just a small block of itself. So, H is locally optimal for I in $[C^0,\infty)$. Consequently, $\mathrm{OPT}(I)\le|\sigma(I)|=\mathrm{H}(I)$. This contradicts the choice of I. □

Lemma 3.2. *$I\setminus\sigma(I)$ does not contain long jobs.*

Proof. Suppose that there is a long job (say J^b) in $I\setminus\sigma(I)$. Since all long jobs are tight and at most one long job is released at one time point, $r(J^b)$ must be equal to the starting time of a short job in H. If J^b is not accepted by OPT, then remove J^b from I without changing the value $\mathrm{H}(I)$ and $\mathrm{OPT}(I)$. Consequently, we can obtain a smaller counterexample, a contradiction. Suppose that OPT accepts J^b at time $r(J^b)$. Note that, $U(r(J^b))$ is not a complete job set at time $r(J^b)$, i.e., EDD strategy cannot accept all jobs in $U(r(J^b))$. This means that at least one short job from $U(r(J^b))$ cannot be accepted by OPT after time $r(J^b)+k$. Since all jobs with kind release times, all short jobs in $U(r(J^b))$ with release times equal to or less than $r(J^b)$. Suppose $J^a\in U(r(J^b))$ is not accepted by OPT, then we can construct a new optimal schedule by removing J^b from I and scheduling J^a at time $r(J^b)$. Since $\mathrm{H}(I\setminus\{J^b\})=\mathrm{H}(I)$, the new instance $I\setminus\{J^b\}$ is a smaller counterexample, a contradiction. □

By Lemma 3.1, we can see that $\sigma(I)$ contains both big blocks and small blocks. Set $a_j=|\sigma_j^a(I)|$ and $b_j=|\sigma_j^b(I)|$, where $j=0,1,...,m$. Then $\mathrm{H}(I)=\alpha+\beta=\sum_{0\le j\le m}(a_j+b_j)$. Since all jobs in $\sigma(I)$ have been completed by H at time C^I and by Observation 3.2, all jobs from $I\setminus\sigma(I)$ expire at time C^I and $[C^0,C^I)=(\cup_{0\le j\le m}[S_j^b,C_j^b))\bigcup(\cup_{0\le j\le m}[S_j^a,C_j^a))$.

For each $0\le j\le m$, we define the job sets $Q_j=\{J^a\in\sigma_j^a(I):r(J^a)<S_j^a\}\cup\{J^a\in I\setminus\sigma(I):r(J^a)<S_j^a$ and $d(J^a)\in[S_j^a+1,C_j^a+1)\}$ and $R_j=\{J^a\in\sigma_j^a(I):r(J^a)\ge S_j^a\}\cup\{J^a\in I\setminus\sigma(I):r(J^a)\ge S_j^a$ and $d(J^a)\in[S_j^a+1,C_j^a+1)\}$. Then $(Q_j\cup R_j)\subset X_j$. By Observation 3.3, $\bigcup_{0\le j\le m}(Q_j\cup R_j)$ contains all the short jobs from I. Let π_1 and π_2 denote the optimal schedules for Q_j and R_j, respectively. And let u_j^* and v_j^* denote the number of jobs accepted by π_1 and π_2 for Q_j and R_j, respectively. Then we have the following lemma.

Lemma 3.3. *$u_j^*\le a_j$ and $v_j^*\le a_j$, $0\le j\le m$.*

Proof. We first prove that $u_j^*\le a_j$ for $0\le j\le m$. Let $\pi_1(Q_j)$ denote the set of jobs accepted in π_1 for Q_j and J^b denote the last long job in $\sigma_j^b(I)$. Then $|\pi_1(Q_j)|=u_j^*$ and $c(J^b)=S_j^a$. For any job $J^a\in Q_j$, by the definition of Q_j, we have $r(J^a)<S_j^a$. Since all jobs have kind release times, we have $r(J^a)\le s(J^b)$. So, all jobs from Q_j are unscheduled effective jobs available at time $s(J^b)$ in H. Namely, $\pi_1(Q_j)\subseteq Q_j\subseteq U(s(J^b))$. Note that H picks J^b to start at time $s(J^b)$. This implies that $U(s(J^b))$ is a complete job set at time $s(J^b)$. So all jobs in $\pi_1(Q_j)$ can be processed on time in $[S_j^a,\infty)$ under EDD ruler. Set $X=\sigma_j^a(I)\cup\{J^a\in I\setminus\sigma(I):d(J^a)\in[S_j^a+1,C_j^a+1)\}\cup\{J^b\in I:r(J^b)\in[S_j^a,C_j^a)\}$. Then $\pi_1(Q_j)\subseteq Q_j\subseteq X_j$. Since H is locally optimal in $[S_j^a,\infty)$, the number of jobs accepted by OPT for X_j in $[S_j^a,\infty)$ cannot be larger than $|\sigma_j^a(I)|=a_j$. Hence, we have $u_j^*\le a_j$.

By the definition of R_j, all jobs in $R_j \subseteq X_j$ released at (or after) time S_j^a. Since H is locally optimal for X_j in $[S_j^a, \infty)$, we have $v_j^* \le a_j$. The result holds. □

For each $(Q_j \cup R_j) \subset X_j$ with $0 \le j \le m$, since H is locally optimal for X_j in $[S_j^a, \infty)$, we can obtain that the number of jobs accepted by OPT for $Q_j \cup R_j$ in $[S_j^a, \infty)$ cannot be larger than a_j. So, the number of short jobs from I accepted by OPT in the time period $[S_0^a, C_0^a) \cup [S_1^a, C_1^a) \cup \cdots \cup [S_{m-1}^a, C_{m-1}^a) \cup [S_m^a, \infty)$ cannot be larger than $a_0 + a_1 + \cdots + a_m = \alpha$. Moreover, for each time interval $[S_j^b, C_j^b)$, since $|\sigma_j^b(I)| = b_j$ and the length of short jobs is 1, there are at most $\lceil b_j k \rceil$ short jobs can be started by OPT in $[S_j^b, C_j^b)$, $j = 1, 2, ..., m$.

Lemma 3.4. *There are at most* $\min\{\beta\lceil k \rceil, \alpha\} + \alpha$ *short jobs from* I *can be accepted by OPT in* $[C^0, \infty)$.

Proof. If $\beta\lceil k \rceil > \alpha$, then $\min\{\beta\lceil k \rceil, \alpha\} + \alpha = 2\alpha$. Note that $\bigcup_{0 \le j \le m}(Q_j \cup R_j)$ contains all the short jobs in I. By Lemma 3.3, we conclude that there are at most 2α short jobs accepted by OPT in $[C^0, \infty)$. Suppose that $\beta\lceil k \rceil \le \alpha$. For each $0 \le j \le m$, since the number of short jobs accepted by OPT in $[S_j^b, C_j^b)$ cannot be larger than $\lceil b_j k \rceil$, and by the fact that $\sum_{0 \le j \le m} \lceil b_j k \rceil \le \lceil k \rceil \sum_{0 \le j \le m} b_j = \beta\lceil k \rceil$, there are at most $\beta\lceil k \rceil$ short jobs from I that can be accepted by OPT in $[C^0, C_0^b) \cup [S_1^b, C_1^b) \cup \cdots \cup [S_m^b, C_m^b)$. Moreover, the number of short jobs accepted by OPT in $[S_0^a, C_0^a) \cup [S_1^a, C_1^a) \cup \cdots \cup [S_{m-1}^a, C_{m-1}^a) \cup [S_m^a, \infty)$ cannot be larger than α. So, the number of short jobs accepted by OPT in $[C^0, \infty)$ cannot be larger than $\beta\lceil k \rceil + \alpha = \min\{\beta\lceil k \rceil, \alpha\} + \alpha$. The result holds. □

We distinguish the following two cases to prove that the smallest counterexample I does not exist.

Case 1. $\beta\lceil k \rceil \le \alpha$. By Lemma 3.4, there are at most $\beta\lceil k \rceil + \alpha$ short jobs accepted by OPT in $[C^0, \infty)$. Since all long jobs have tight deadline, by Lemma 3.3, and from the fact that, the number of short jobs from I accepted by OPT in the time period $[S_0^a, C_0^a) \cup [S_1^a, C_1^a) \cup \cdots \cup [S_{m-1}^a, C_{m-1}^a) \cup [S_m^a, \infty)$ cannot be larger than α, we have $\mathrm{OPT}(I) \le \beta\lceil k \rceil + \alpha$. So, it follows that

$$\frac{\mathrm{OPT}(I)}{\mathrm{H}(I)} \le \frac{\beta\lceil k \rceil + \alpha}{\alpha + \beta} \le \frac{2\lceil k \rceil}{1 + \lceil k \rceil} = \rho,$$

where the last inequality holds since $\beta\lceil k \rceil \le \alpha$. This contradicts the fact that I is the smallest counterexample.

Case 2. $\beta\lceil k \rceil > \alpha$. Let α^* denote the number of short jobs from I accepted by OPT in $[C^0, \infty)$. By Claim Lemma 3.4, we have $\alpha^* \le 2\alpha$. Since there are at most α short jobs accepted by OPT in $[S_0^a, C_0^a) \cup [S_1^a, C_1^a) \cup \cdots \cup [S_{m-1}^a, C_{m-1}^a) \cup [S_m^a, \infty)$, there are at least $\alpha^* - \alpha$ short jobs needed to be scheduled in $[R_I, C_0^b) \cup [S_1^b, C_1^b) \cup \cdots \cup [S_m^b, C_m^b)$. Note that the length of long jobs is k. Since all long jobs have tight deadlines, and by Lemma 3.3, there are at most $\beta - \lceil \frac{\alpha^* - \alpha}{\lceil k \rceil} \rceil$ long jobs that can be accepted by OPT for I. Consequently, $\mathrm{OPT}(I) \le \beta - \lceil \frac{\alpha^* - \alpha}{\lceil k \rceil} \rceil + \alpha^* \le \beta - (\alpha^* - \alpha)/\lceil k \rceil + \alpha^*$. It follows that

$$\frac{\mathrm{OPT}(I)}{\mathrm{H}(I)} \le \frac{\beta - (\alpha^* - \alpha)/\lceil k \rceil + \alpha^*}{\alpha + \beta} = \frac{\beta + \alpha/\lceil k \rceil + (1 - 1/\lceil k \rceil)\alpha^*}{\alpha + \beta}$$

$$\leq \frac{\beta + 2\alpha - \alpha/\lceil k\rceil}{\alpha + \beta} = \frac{\beta/\alpha + 2 - 1/\lceil k\rceil}{1 + \beta/\alpha} < \frac{2\lceil k\rceil}{\lceil k\rceil + 1} = \rho,$$

where the second inequality and the last inequality hold since $\alpha^* \leq 2\alpha$ and $\beta\lceil k\rceil > \alpha$, respectively. This still contradicts the fact that I is the smallest counterexample.

By summing up the above discussion, we conclude the main result of this section as follows.

Theorem 3.2. *For problem* $1 \mid online, KRT, r(J), p(J) \in \{1, k\} \mid \sum E(J)$*, when all long jobs are tight, H has a competitive ratio of* $2\lceil k\rceil/(\lceil k\rceil + 1)$.

It is easy to find an instance to prove that the bound $2\lceil k\rceil/(\lceil k\rceil + 1)$ is tight. For example, we construct a job sequence I with $2\lceil k\rceil$ unit length jobs and one k length job. Let $\mathcal{J}_1 = \{J_j : 0 \leq j \leq \lceil k\rceil\}$ denote the set of the first $\lceil k\rceil + 1$ jobs, such that $r(J_j) = 0$, $p(J_j) = 1$, $d(J_j) = 2\lceil k\rceil$, $j = 1, 2, ..., \lceil k\rceil$. In particular, when $j = 0$, suppose that $p(J_0) = d(J_0) = k$. Let $\mathcal{J}_2 = \{J'_j : 1 \leq j \leq \lceil k\rceil\}$ denote the set of the other $\lceil k\rceil$ jobs, such that $r(J'_j) = k$, $p(J'_j) = 1$, $d(J'_j) = 2\lceil k\rceil$, $j = 1, 2, ..., \lceil k\rceil$. Then H will accept $\lceil k\rceil$ short jobs from $\mathcal{J}_1 \cup \mathcal{J}_2$ and one long job J_0. While OPT will accept all short jobs in $\mathcal{J}_1 \cup \mathcal{J}_2$. So, $\text{OPT}(I)/\text{H}(I) = 2\lceil k\rceil/(\lceil k\rceil + 1)$.

Remark 3.1. Note that $2\lceil k\rceil/(\lceil k\rceil + 1) = 4/3$ if $1 < k \leq 2$. Then by Theorem 3.1, algorithm H is a best possible online algorithm for this problem.

§4 For the objective function: $\sum p(J)E(J)$

In this section, we study the problems $1 \mid \text{online}, \text{KRT}, r(J) \mid \sum p(J)E(J)$, $1 \mid \text{online}, \text{KRT}, r(J), p(J) \in \{1, k\} \mid \sum p(J)E(J)$, and $1 \mid \text{online}, \text{KRT}, r(J), p(J) \in \{1, k\} \mid \sum p(J)E(J)$ under the constraint that all the k length jobs are tight. When no confusion can occur, we use $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ to denote the above three problems, respectively.

4.1 The lower bounds

Here, we give a lower bound of $\max\{4/(2 + k), 1\}$ for $\mathcal{P}_3$, a lower bound of $\min\{(1 + k)/k, 2k/(1 + k)\}$ for $\mathcal{P}_2$, and a lower bound of $\sqrt{2}$ for $\mathcal{P}_1$, respectively. By constructing the same job instance as in Theorem 3.1 and using the similar proof scheme, we conclude the following theorem.

Theorem 4.1. *For problem* $\mathcal{P}_3$*, no online algorithms have a competitive ratio less than* $\max\{4/(2 + k), 1\}$.

Theorem 4.2. *For problem* $\mathcal{P}_1$ *and* $\mathcal{P}_2$*, no online algorithms have a competitive ratio less than* $\sqrt{2}$ *and* $\min\{(1 + k)/k, 2k/(1 + k)\}$*, respectively.*

Proof. Set $\alpha = 1 + \sqrt{2}$ (for $\mathcal{P}_1$), $\alpha = k$ (for $\mathcal{P}_2$). Consider an online algorithm $\mathcal{A}$ and the following instance I presented by the adversary.

At time 0, there are two jobs J_1 and J_2 with $p(J_1) = 1 = d(J_1)$, $p(J_2) = \alpha$, $d(J_2) = 1 + \alpha$ arrive. If algorithm $\mathcal{A}$ starts J_1 at time 0, then jobs J_3 with $p(J_3) = p(J_2) = \alpha$, $d(J_3) = 2\alpha$

arrives at time 1. Since algorithm $\mathcal{A}$ can only schedule one long job during the time interval $[1, 2\alpha]$ by $\mathcal{A}$, $\mathcal{A}(I) \leq 1+\alpha$. Note that OPT can schedule two long jobs in $[0, 2\alpha]$. It follows that

$$\mathrm{OPT}(I)/\mathcal{A}(I) \geq \frac{2\alpha}{1+\alpha}(= \alpha - 1 = \sqrt{2} \text{ for } \mathcal{P}_1).$$

If algorithm $\mathcal{A}$ does not start J_1 at time 0, then no other jobs will arrive, and so, $\mathcal{A}(I) \leq p(J_2) = \alpha$ and $\mathrm{OPT}(I) = p(J_1) + p(J_2) = 1 + \alpha$. Hence,

$$\mathrm{OPT}(I)/\mathcal{A}(I) \geq \frac{1+\alpha}{\alpha}(= \alpha - 1 = \sqrt{2} \text{ for } \mathcal{P}_1).$$

The result follows. □

4.2 Online algorithm LEJ

In this subsection, we present an online algorithm LEJ (Longest Effective Jobs) for $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$. Let t be the current time. We use $U(t)$ to denote the set of all unscheduled effective jobs at time t. Online algorithm LEJ works as follows.

Algorithm LEJ: At any time t, if $U(t) \neq \emptyset$ and the machine is idle, then choose a job with the longest processing time in $U(t)$ (In case of ties, we first choose a job with the smallest deadline, and then a job with the smallest release time if there is still a choice) and process it on the machine; Otherwise do nothing and merely increment t.

4.3 Analysis

In this subsection, we show that the competitive ratio of LEJ for problems $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ is $\varrho_1 = 3$, $\varrho_2 = 1 + \lceil k \rceil / k$, and $\varrho_3 = \lceil k \rceil / k$, respectively. Let I_i be a smallest counterexample about $\mathcal{P}_i$, $i = 1, 2, 3$. Then $\mathrm{OPT}(I_i) > \varrho_i \mathrm{LEJ}(I_i)$. Let σ_i denote the schedule produced by LEJ for instance I_i, $i = 1, 2, 3$.

By shifting, we may assume that the release time of the first job in I_i $(i = 1, 2, 3)$ is 0. Let S_0 and C_0 denote the starting time of the first job and the completion time of the last job in σ_i, respectively. By LEJ, we have $S_0 = 0$. For any job $J \in I_i$, we use $s(J)$, $s^*(J)$, $c(J)$, $c^*(J)$ to denote the starting time and completion time of J in LEJ and OPT, respectively. Let $\sigma_i(I_i)$ be the set of jobs accepted by LEJ for I_i, $i = 1, 2, 3$. The following two observations are implied in the implementation of algorithm LEJ.

Observation 4.1. *All jobs in $I_i \setminus \sigma_i(I_i)$ are expired at time C_0, $i = 1, 2, 3$.*

Observation 4.2. *All jobs in $\sigma_i(I_i)$ are processed contiguously by LEJ, i.e., there is no idle time interval between $[0, C_0)$, $i = 1, 2, 3$.*

If possible, let s^* be the starting time of the last job (say J^*) from $I_1 \setminus \sigma_1(I_1)$ in OPT. By Observation 4.1, we have $s^* < C_0$. So, $\mathrm{OPT}(I_1 \setminus \sigma_1(I_1)) \leq s^* + p(J^*) < C_0 + p(J^*)$. By Observation 4.2, we have $\mathrm{LEJ}(I_1) = \sum_{J \in \sigma_1(I_1)} p(J) = C_0$. We use J' to denote the last job in σ_1. Then $s(J') + p(J') = C_0 = \mathrm{LEJ}(I_1)$. Since all jobs in I_1 have kind release times, we have

$J', J^* \in E(s(J'))$. By LEJ, $p(J^*) \le p(J') \le C_0$. So, we have

$$\begin{aligned} \mathrm{OPT}(I_1) &\le \mathrm{OPT}(\sigma_1(I_1)) + \mathrm{OPT}(I_1 \setminus \sigma_1(I_1)) < \mathrm{LEJ}(I_1) + C_0 + p(J^*) \\ &\le \mathrm{LEJ}(I_1) + C_0 + C_0 = 3\mathrm{LEJ}(I_1) = \varrho_1 \mathrm{LEJ}(I_1). \end{aligned}$$

This contradicts with the assumption that I_1 is a counterexample of $\mathcal{P}_1$. So, we conclude the following theorem.

Theorem 4.3. *For problem* $1 \mid online, KRT, r(J) \mid \sum p(J)E(J)$*, LEJ has a competitive ratio of* 3.

It is easy to find an instance to prove that the bound 3 is asymptotically tight. For example, we construct a job instance I with three jobs as follows: $r(J_1) = r(J_2) = r(J_3) = 0$; $p(J_1) = 1, p(J_2) = 1-\epsilon, p(J_3) = 1+\epsilon$; $d(J_1) = 1, d(J_2) = 2-\epsilon, d(J_3) = 3$, where ϵ is a sufficiently small positive real number. Then LEJ will schedule J_3 at time 0. While OPT can schedule J_1, J_2, J_3 at time $0, 1, 2-\epsilon$, respectively. So, $\frac{\mathrm{OPT}(I)}{\mathrm{LEJ}(I)} = \frac{3}{1+\epsilon} \to 3$ as $\epsilon \to 0$.

Next, we will analyze the competitive ratio of algorithm LEJ for $\mathcal{P}_2$ and $\mathcal{P}_3$. Some useful notations and definitions presented in Section 3.3 will be adopted in the following discussion. We refer to the length 1 jobs as short jobs, and the length k jobs as long jobs. Moreover, a short job is indicated by J^a and a long job is indicated by J^b. Note that, σ_i is the schedule produced by LEJ for the smallest counterexample I_i, $i = 2, 3$. Then we have the following two observations.

Observation 4.3. *Instance I_i must consist of short jobs and long jobs, $i = 2, 3$.*

Proof. Assume that instance I_i only contains one kind jobs (short jobs or long jobs). Since all jobs in I have kind release times and algorithm LEJ schedules jobs under EDD rule, LEJ is an optimal algorithm for I_i. So, we have $\mathrm{LEJ}(I_i) = \mathrm{OPT}(I_i)$. This contradicts the definition of I_i. □

Observation 4.4. *All the long jobs from I_i are accepted by LEJ, i.e., $I_i \setminus \sigma_2(I_i)$ does not contain long jobs, $i = 2, 3$.*

Proof. Suppose that there is a long job (say J) in $I_i \setminus \sigma_i(I_i)$. At any time point t, since algorithm LEJ first chooses the long jobs in $U(t)$ and schedules them under EDD ruler, and from the fact that all jobs in I_i have kind release times, LEJ is an optimal strategy for the long jobs of I_i. So, there must be a long job (say J') in $\sigma_i(I_i)$ scheduled by LEJ during the time interval $[d(J) - k, d(J))$. By LEJ and J, J' have kind release times, we have $d(J') \le d(J) < C_0 + k$ and $J, J' \in E(s(J'))$. Therefore, we can remove J' from I_i without changing the value $\mathrm{LEJ}(I_i)$ and cannot decrease the offline optimal value. Consequently, we can obtain a smaller counterexample, a contradiction. □

Let a and b denote the number of short jobs and long jobs in $\sigma_2(I_2)$, respectively. Then we have $\mathrm{LEJ}(I_2) = \sum_{J \in \sigma_2(I_2)} p(J) = a + kb$. By Observation 4.2 and the definition of C_0, we have $C_0 = bk + a$. By Observation 4.4, all jobs in $I_2 \setminus \sigma_2(I_2)$ are short jobs, and since the fact that the length of short jobs is 1 and $\lceil bk\rceil + a \geq bk + a = C_0$, at most $\lceil bk\rceil + a$ short jobs of $I_2 \setminus \sigma_2(I_2)$ can be started in $[0, C_0)$. Moreover, by Observation 4.1, all short jobs in $I_2 \setminus \sigma_2(I_2)$ expire at time C_0. So, $\mathrm{OPT}(I_2 \setminus \sigma_2(I_2)) \leq a + \lceil bk\rceil$. By the fact that $\mathrm{LEJ}(I_2) = a + bk$, we have

$$\begin{aligned}\mathrm{OPT}(I_2) &\leq \mathrm{OPT}(\sigma_2(I_2)) + \mathrm{OPT}(I_2 \setminus \sigma_2(I_2)) \leq \mathrm{LEJ}(I_2) + a + \lceil bk\rceil \\ &\leq a + bk + a + \lceil bk\rceil = (1 + \tfrac{a+\lceil bk\rceil}{a+bk})\mathrm{LEJ}(I_2) \\ &\leq (1 + \tfrac{\lceil k\rceil}{k})\mathrm{LEJ}(I_2) = \varrho_2 \mathrm{LEJ}(I_2).\end{aligned}$$

This still contradicts the fact that I_2 is the smallest counterexample of $\mathcal{P}_2$. Hence, we have the following theorem.

Theorem 4.4. *For problem* $1 \mid online, KRT, r(J), p(J) \in \{1, k\} \mid \sum p(J)E(J)$*, LEJ has a competitive ratio of* $1 + \lceil k\rceil/k$.

It is easy to find an instance to prove that the bound $1 + \lceil k\rceil/k$ is tight. For example, we construct a job sequence I with $\lceil k\rceil$ unit length jobs and one k length job. At time 0, one k length job J^* with $d(J^*) = k + \lceil k\rceil$, and the other $\lceil k\rceil$ unit length jobs have the same deadline of $\lceil k\rceil$ arrive. Then LEJ only accept J^*, while OPT will accept all jobs in I. Hence $\frac{\mathrm{OPT}(I)}{\mathrm{LEJ}(I)} = \frac{k+\lceil k\rceil}{k} = 1 + \frac{\lceil k\rceil}{k}$.

Let x and y denote the number of short jobs and long jobs in $\sigma_3(I_3)$, respectively. For convenience, we use $\sigma(I_3)$ to denote $\sigma_3(I_3)$. Then we have $\mathrm{LEJ}(I_3) = \sum_{J\in\sigma(I_3)} p(J) = x + yk$. By Observation 4.2, all jobs in $\sigma(I_3)$ are scheduled continuously. We divide the whole schedule $\sigma(I_3)$ into two sorts of blocks, big blocks $\sigma_j^b(I_3)$ and small blocks $\sigma_j^a(I_3)$ with $j = 0, 1, ..., n$. Then we have the following lemma.

Lemma 4.1. *(i)* $\bigcup_{0\leq j\leq n} \sigma_j^a(I_3) \neq \emptyset$ *and* $\bigcup_{0\leq j\leq n} \sigma_j^b(I_3) \neq \emptyset$*; (ii)* $\sigma_0^b(I_3) \neq \emptyset$*; (iii)* $n = 0$*, i.e.,* $\sigma(I_3) = \sigma_0^b(I_3) \bigcup \sigma_0^a(I_3)$.

Proof. We first prove statement (i). If $\bigcup_{0\leq j\leq n} \sigma_j^b(I_3) = \emptyset$, then by algorithm LEJ, instance I_3 contains only short jobs. This contradicts Observation 4.3. If $\bigcup_{0\leq j\leq n} \sigma_j^a(I_3) = \emptyset$, then all jobs accepted by H for I_3 are long jobs. By Observation 4.2, we have $\mathrm{LEJ}(I_3) = yk = C_0$. By Observation 4.1 and all long jobs of I_3 have kind release times, at most $\lceil yk\rceil$ short jobs can be started by OPT in $[0, C_0)$. So, we have $\mathrm{OPT}(I_3) \leq \lceil yk\rceil$. It follows that $\mathrm{OPT}(I_3)/\mathrm{LEJ}(I_3) \leq \lceil yk\rceil/yk \leq \lceil k\rceil/k = \varrho_3$. This contradicts the choice of I_3. Statement (i) follows.

For statement (ii), suppose to the contrary that $\sigma_0^b(I_3) = \emptyset$. Set $W = U(C_0^a) \cup \{J \in I_3 : r(J) > C_0^a\}$. Note that, all unscheduled jobs in $I_3 \setminus W$ by LEJ expire at time C_0^a. Let π be an optimal schedule of instance I_3. Since all jobs in I_3 have kind release times, the jobs from $U(C_0^a)$ are released at time C_0^a or with release times less than $C_0^a - 1$. Set $U^*(C_0^a) = \{J \in U(C_0^a) : r(J) \leq C_0^a - 1\}$. Since all long jobs in I_3 are tight and by the definition of $U^*(C_0^a)$, all jobs from $U^*(C_0^a)$ are short jobs. Let C_π^* denote the completion time of the last job in π accepted before time C_0^a. Since all jobs in $\sigma_3(I_3)$ scheduled before time C_0^a are short jobs and by LEJ, no long jobs arrive before time C_0^a. So, we have $C_\pi^* < C_0^a + 1$. We define two smaller job instances V^* and W^* by the following way.

Instance V^* consists of the jobs in $I_3 \setminus W$ and the jobs in $U^*(C_0^a)$ with the deadlines of jobs $J^a \in U^*(C_0^a)$ being revised by $d^*(J^a) = C_\pi^*$. Note that, in instance V^*, all jobs in $U^*(C_0^a)$ expire at time C_0^a since $C_\pi^* < C_0^a + 1$. Furthermore, algorithm LEJ accepts the same number of short jobs in time interval $[0, C_0^a)$ for both instances I_3 and V^*. So, we have $\text{LEJ}(V^*) = |\sigma_0^a(I_3)|$.

Instance W^* consists of the jobs in $U(C_0^a) \cup \{J : r(J) > C_0^a\}$ with the release dates of jobs $J \in U(C_0^a)$ being revised by $r^*(J) = C_0^a$. Note that all jobs in W^* have release dates at least C_0^a. Furthermore, algorithm LEJ generates the same schedule in time interval $[C_0^a, \infty)$ for both instances I_3 and W^*. So, we have $\text{LEJ}(W^*) = x + yk - |\sigma_0^a(I_3)|$.

By the definition of V^* and W^*, we can see that LEJ accepts the same total processing time of jobs for I_3 and $V^* \cup W^*$. Then we have $\text{LEJ}(V^* \cup W^*) = \text{LEJ}(I_3)$. Furthermore, the schedules $\pi(V^*)$ and $\pi(W^*)$ obtained from π by restricting π on the jobs accepted in $[0, C_0^a)$ and $[C_0^a, \infty)$, respectively, can be regarded as feasible schedules of V^* and W^*, respectively. So, $\text{OPT}(I_3) \le \text{OPT}(V^*) + \text{OPT}(W^*)$. By the choice of I_3, we have $\text{OPT}(V^*) \le \varrho_3\text{LEJ}(V^*)$ and $\text{OPT}(W^*) \le \varrho_3\text{H}(W^*)$. Thus, we have

$$\text{OPT}(I_3) \le \text{OPT}(V^*) + \text{OPT}(W^*) \le \varrho_3(\text{LEJ}(V^*) + \text{LEJ}(W^*)) = \varrho_3\text{LEJ}(V^* \cup W^*) = \varrho_3\text{LEJ}(I_3).$$

This contradicts with the assumption that I_3 is a counterexample of $\mathcal{P}_3$.

If $n = 0$, by Statements (i) and (ii), we have $\sigma(I_3) = \sigma_0^b(I_3) \bigcup \sigma_0^a(I_3)$. Assume that $n \ge 1$. Set $W' = U(C_0^a) \cup \{J \in I_3 : r(J) > C_0^a\}$. Set $U^*(C_0^a) = \{J \in U(C_0^a) : r(J) \le C_0^a - 1\}$. Since all long jobs in I_3 are tight, and by Observation 4.4, all jobs from $U^*(C_0^a)$ are short jobs. Note that all long jobs in $\sigma_0^b(I_3)$ with release times less than $C_0^b - k$ and they are all tight jobs. These long jobs form $\sigma_0^b(I_3)$ cannot be started after time $C_0^b - k$ in any optimal schedule. So, for each long job J^b with $r(J^b) < C_0^a$, $c^*(J^b) \le C_0^b < C_0^a < C_0^a + 1$; For each short job J^a with $r(J^a) < C_0^a$, $c^*(J^a) < C_0^a + 1$. Consequently, we have $C_\pi^* < C_0^a + 1$, where C_π^* denote the completion time of the last job in π (optimal schedule) accepted before time C_0^a. Using the same proof scheme as in Statement (ii), we can prove that Statement (iii) is correct. □

Lemma 4.1 (iii) implies that all long jobs are scheduled before short jobs in LEJ, i.e., $C_0^b = S_0^a$ and $C_0^a = C_0$. Set $X = \sigma_0^a(I_3) \cup (I_3 \setminus \sigma(I_3))$. By Observation 4.4, all jobs from X are short jobs. From the execution of algorithm LEJ, we know that LEJ schedules these short jobs in $\sigma_0^a(I_3)$ under EDD strategy. Since all jobs in I_3 have kind release times, LEJ is an optimal strategy for the job set X in the time interval $[S_0^a, \infty)$. So, the number of jobs accepted by OPT in $[S_0^a, \infty)$ cannot be larger than $|\sigma_0^a(I_3)| = x$. Recall that $|\sigma_0^b(I_3)| = y$ and the length of short jobs is 1. Thus at most $\lceil yk \rceil$ short jobs can be started by OPT in $[0, S_0^a)$. So, $\text{OPT}(I_3) \le x + \lceil yk \rceil$. By the fact that $\text{LEJ}(I_3) = x + yk$, we have

$$\frac{\text{OPT}(I_3)}{\text{LEJ}(I_3)} \le \frac{x + \lceil yk \rceil}{x + yk} \le \frac{\lceil k \rceil}{k} = \varrho_3.$$

This contradicts with the assumption that I_3 is a counterexample of $\mathcal{P}_3$. Hence, we conclude the following theorem.

Theorem 4.5. *For problem $1 \mid online, KRT, r(J), p(J) \in \{1, k\} \mid \sum p(J)E(J)$, when all long jobs are tight, LEJ has a competitive ratio of $\lceil k \rceil / k$.*

Remark 4.1. Note that $\lceil k \rceil / k = 1$ if k is any positive integer. Then by Theorem 4.1, algorithm LEJ is an optimal online algorithm for $\mathcal{P}_3$.

References

[1] E J Anderson, C N Potts. *Online scheduling of a single machine to minimize total weighted completion time*, Mathematics of Operations Research, 2004, 29: 686-697.

[2] K R Baker. *Introduction to Sequencing and Scheduling*, John Wiley & Sons, New York, 1974.

[3] S K Baruah, J Haritsa, N Sharma. *On-line scheduling to maximize task completions*, Journal of Combinatorial Mathematics and Combinatorial Computing, 2001, 39: 65-78.

[4] M Chrobak, W Jawor, J Sgall, T Tichý. *Online Scheduling of Equal-Length Jobs: Randomization and Restarts Help*, SIAM Journal on Computing, 2007, 36: 1709-1728.

[5] S A Goldman, J Parwatikar, S Suri. *On-line scheduling with hard deadlines*, Journal of Algorithms, 2000, 34: 370-389.

[6] H Hoogeveen, C N Potts, G J Woeginger. *On-line scheduling on a single machine: Maximizing the number of early jobs*, Operations Research Letters, 2000, 27: 193-196.

[7] P Keskinocak. *Online algorithms with lookahead: A survey*, ISYE working paper, 1999.

[8] P H Liu, X W Lu, Y Fang. *A best possible deterministic on-line algorithm for minimizing makespan on parallel batch machines*, Journal of Scheduling, 2012, 15: 77-81.

[9] W J Li, J J Yuan. *LPT online strategy for parallel-machine scheduling with kind release times*, Optimization Letters, 2016, 10: 159-168.

[10] K Pruhs, J Sgall, E Tong. *Online scheduling*, in: J Y-T Leung (Eds), Handbook of Scheduling: Algorithm, Model, and Pertormance Analysis, Chapman & Hall/CRC Press, Boca Raton, FL, USA, 2004.

[11] Z Y Tan, A Zhang. *Online and semi-online scheduling*, in: P M Pardalos, et al (Eds), Handbook of Combinatorial Optimization, New York, Springer, 2013.

[12] J Tian, T C E Cheng, C T Ng, J J Yuan. *Online scheduling on unbounded parallel-batch machines to minimize the makespan*, Information Processing Letters, 2009, 109: 1211-1215.

[13] J Tian, R Y Fu, J J Yuan. *Online over time scheduling on parallel-batch machines: a survey*, Journal of the Operations Research Society of China, 2014, 2: 445-454.

[1] School of Mathematical Sciences, Luoyang Normal University, Luoyang 471934, Henan, People's Republic of China. Email: liwenjie0521@163.com

[2] School of Mathematics and Statistics, Qingdao University, Qingdao 266071, Shandong, People's Republic of China.

[3] Department of Information and Computation Science, Zhongyuan University of Technology, Zhengzhou 450007, Henan, People's Republic of China.