A Near-Optimal Semi-Online Algorithm for Maximizing Throughput Scheduling Problem [*]

Ye Tao [†] Zhijun Chao [‡] Yugeng Xi [§]

Abstract—**This paper presents a new kind of models defined as $1|r_j, r_t^1, \mathfrak{C}|\sum C_j$, which covers a wide range of models in a real-time system such as periodic, aperiodic and sporadic task models. We provide a semi-online algorithm AOPT for such a model with a reasonable assumption. In its worst-case analysis, we demonstrate that the lower bound of the competitive ratio by AOPT equals to 1.5387 while the upper bound is 1.57, which gives rise to a narrow gap of 0.0313. In addition, we generate and utilize a novel proving technique in the process of upper bound analysis, the essence of which is to transform an arbitrary instance to the one with a computable performance ratio. Through such process, we argue that the introduction of some future information will improve the performance of an algorithm.**

Keywords: scheduling, semi-online algorithm, worst-case analysis, competitive ratio, real-time system

1 Introduction

One of the functions of a multitasking computer operating system is to schedule the time that the CPU devotes to the different programs that have to be executed [1]. We focus on our attention in this paper to the uniprocessor scheduling. The real-time system, $\mathcal{A}$, is modeled as being comprised of a number of concurrent tasks (or threads) that are featured by several parameters. Each task τ_i gives rise to a series of jobs that are to be executed on a single processor. Each job j is characterized by (r_j, p_j) where r_j means the release time of job j and p_j equals to the processing time of such a job. Usually, a system can be classified into periodic system, aperiodic system or sporadic system. We show that no matter which system it is, the competitive ratio of instances in $\mathfrak{C}$ by the semi-online algorithm AOPT we designed is between 1.5387 and 1.57.

Great progress has been made in over 10 years. [2] purposes an optimal online algorithm D-SPT for a single machine problem with a $2-$competitiveness in 1996. [3] presents a naive greedy algorithm in computer science background which always schedules the heaviest job is known to be 2-competitive in 2000. [4] presents an optimal online algorithm for another single machine problem with a $1.618-$competitiveness in 2000. [5] designs an optimal online algorithm for packet scheduling with agreeable deadlines. Their research background lies in IP-based Qos networks and the algorithm has a $\phi-$competitive ratio. [6] discuss the online competitive algorithms for maximizing weighted throughput of unit jobs and gives a random algorithm which is approximately 1.582-competitive. The latest result is from [7] who gives a semi-online algorithm α-PSNR which is 1.707-competitive.

In Section 2, we first introduce some models in a background of computer science and show that they can be deduced into a single machine scheduling problem. Then we purpose a new semi-online algorithm AOPT in Section 3 and present the basic assumption of the instance-space $\mathfrak{C}$. Afterwards, in Section 4 we mainly demonstrate the lower bound of such algorithm is 1.5387 and utilize a novel technique to show that the upper bound of AOPT is 1.57. All efforts show the power of future information, which improves the best existing result of 2 in [2] and 1.7071 in [7]. Finally, we provide the conclusion in Section 5.

2 Preliminaries

In a real time system $\mathcal{A}$, a basic assumption of any deterministic algorithm is that all information is known to the scheduler in advance. However, the practical situation always breaks this assumption. In most cases, we could only have the knowledge of past and current information about jobs. We name such algorithm by *online* algorithm. Nevertheless, with the increasing demand of performance of an algorithm, a new kind of algorithm called *semi-online* algorithm appears. Since the information of semi-online algorithm is more than online one, we intuitively have an idea that the performance of semi-online algorithm should be 'better' than that of online algorithm.

We are now ready to give the definition of the concept of '*ϕ-competitive*' and '*competitive ratio*', which both describe the performance of an online or semi-online algorithm in a worst-case view.

[*]This work is supported by the National Science Foundation of China (No.60504026, 60674041) and Specialized Research Fund for the Doctoral Program of High Education(No. 20070248004)

[†]Corresponding author, Department of Automation, Shanghai Jiao Tong University, Shanghai, PRC. Email: issuetina@sjtu.edu.cn

[‡]Email: zhijun.chao@gmail.com

[§]Email: ygxi@sjtu.edu.cn

Definition 1. *Let $\mathbb{A}$ be an online or semi-online algorithm. For any instance I in the whole instance-space $\mathbb{I}$, let φ be the schedule by $\mathbb{A}$, and π be the optimal schedule which might be unknown due to the NP-hardness of the problem. Suppose that $f(\cdot)$ is the objective function. Then the* ***competitive ratio*** *ρ holds that*

$$\rho = \max_{I \in \mathbb{I}} \left\{ \frac{f(\varphi(I))}{f(\pi(I))} \right\}.$$

*Moreover, if there is a ϕ satisfying $f(\varphi(I)) \le \phi \cdot f(\pi(I))$ for any instance $I \in \mathbb{I}$, then the algorithm $\mathbb{A}$ is ϕ-***competitive***. Obviously, $\rho \le \phi$.*

According to [6], a system is *schedulable* with respect to a specified scheduling policy if it will meet all its timing requirements when executed on its target platform with that scheduling policy. Also, [6] defines the concept of *sustainability*, which is the subset of schedulable systems. These notions are well-understood in the real time systems community. Different task models place different restrictions on the parameters of the sequences of jobs. Most popular models are listed as follows: *periodic task models* specifying that successive jobs arrive at an exact pre-specified time, *aperiodic task models* specifying that successive jobs arrive at an unfixed time, and *sporadic task models* specifying a minimum temporal separation between the arrivals of successive jobs of a task.

Theorem 1. *For any instance I in a real time system $\mathcal{A}$, no matter it is schedulable, sustainable, periodic, aperiodic, sporadic or not, I could be regarded as an instance in a single machine scheduling problem [1].*

For ease of discussion, some assumptions will be placed on the problems. In literatures, [6] and [8] assume that the jobs are all unit jobs because of the metered-task model . [9] purposes the assumption of decreasing size jobs which means $p_{j+1} \le p_j$ for $r_{j+1} \ge r_j$. [7] raises the assumption that the next release time $r_t^1 \triangleq \min\{r_j | r_j > t\}$ is known in advance. In this paper, we also give an assumption with respect to the notion of $\mathfrak{C}$ with an Ω_t which describes the available job set where all jobs released no later than t and haven't been scheduled yet.

Assumption 1. *Assume that instance-space $\mathfrak{C}$ is constructed as follows. For any instance $I \in \mathfrak{C}$, if we define $p_{(j)}$ as the minimum processing time job in Ω_{r_j} and s_j as the starting time of $p_{(j)}$, then there must be an integer k such that*

$$1+\alpha < \frac{\tilde{C}_{(j)}}{r_t^1} < +\infty \quad ,1 \le j \le k$$

$$1 \leqslant \frac{\tilde{C}_{(j)}}{r_t^1} \leqslant 1+\alpha \qquad ,j > k$$

where $\tilde{C}_{(j)}$ represents the virtual completion time of $p_{(j)}$, ie., $\tilde{C}_{(j)} = r_j + p_j$ for all $1 \le j \le k$ while $\tilde{C}_{(j)} = s_j + p_j$ for all $j > k$. And $\alpha \in [0.5, 1)$ is a parameter of the algorithm $\mathbb{A}$.

3 System Model and Algorithm AOPT

The basic model of $1|r_j|\sum C_j$ can be found in [1]. Although the optimal algorithm for $1|r_j|\sum C_j$ is SPT [10] when jobs all arrive at the identical time, literature [11] shows such problem is strongly NP-hard when all jobs arrive arbitrarily.

In this paper, we discuss an extended model of $1|r_j, r_t^1, \mathfrak{C}|\sum C_j$, which means the next release time r_t^1 is known in advance and the instance-space $\mathfrak{C}$ fits the assumption 1. Afterwards, we provide a new semi-online algorithm AOPT as follows.

```
ALGORITHM AOPT (Almost-Optimal)
STEP 0.  If the machine is idle and a job is available
           at time $t$, determine available job set $\Omega_t$,
           $j = \arg\min\{p_j | j \in \Omega_t\}$ and $r_t^1$;
         Otherwise, wait until the machine is idle
           and a job is available.
STEP 1.  If $r_t^1 < +\infty$, then
         STEP 1A. if $t + p_j \le (1+\alpha)r_t^1$,
                    schedule job $j$
                    and let $\Omega_t = \Omega_t - \{j\}$.
         STEP 1B. if $t + p_j > (1+\alpha)r_t^1$,
                    go to STEP 0.
STEP 2.  Go to STEP 0.
```

Intuitively, the algorithm whose scheduled jobs are more than others could be 'better' than others. For instance, we know that the optimal online-algorithm D-SPT [2] has $\rho = 2$. Moreover, Chao [7] shows that α-PSNR semi-online algorithm owns $1.5387 \le \rho \le 1 + \sqrt{2} \approx 1.7071$, which improves the result of [2]. We conjecture our algorithm AOPT is better than both [2] and [7]. We further demonstrate our conjecture in the following section 4.

4 Worst-case Analysis of AOPT

We will prove that AOPT has worst-case ratio no less than 1.5387 and no more than 1.57. In our proof, we work with a worst-case analysis in which the worst-case has and only has one contiguously processing block.

It is trivial to set $r_1 = 0$. Since $I \in \mathfrak{C}$, it is obvious that the jobs, no matter the number of them is finite or infinite, will be scheduled in a continuously processing block by AOPT. Thus, we obtain the following lemma.

Lemma 1. *For any instance $I \in \mathfrak{C}$, it consists of a single block*[1] *by AOPT.*

4.1 Lower Bound Analysis

Theorem 2 (Lower Bound). *For the system model $1|r_j, r_t^1|\sum C_j$, the lower bound of the competitive ratio ρ by any semi-online algorithm $\mathbb{A}$ is* 1.5387.

[1] A single block: it starts with a nonnegative period of idle time after which all jobs are executed contiguously.

Proof. In order to demonstrate the conclusion, we construct two kinds of instances for any semi-online algorithm $\mathbb{A}$ with r_t^1 to attain a large enough lower bound of ρ.

Assume that both kinds are in accordance with the condition that the number of jobs is no less than 2 and $r_{j+1} = r_2(1 + r_j)$ ie. $r_{j+1} = \frac{r_2(r_2^j - 1)}{r_2 - 1}$ where $j \in \mathbb{N}$. Since for any time t, it meets that $t \in [r_j, r_{j+1})$ and either the job will be scheduled at time t or else with regards to any $\mathbb{A}$.

1. If $\exists j \in \mathbb{N}$ such that $t \in [r_j, r_{j+1})$ is the earliest starting time according to any fixed algorithm and the scheduled job is a unit processing time job. We have known the next release time $r_t^1 = r_{j+1}$ at time t. So we create a new release time $r_{j+1} + \varepsilon$ and n zero processing time jobs at that time. Thus, we have $C(\varphi) = (n+1)(t+1)$ and $C(\psi) = (n+1)(r_{j+1}+\varepsilon)+1$. Let the number of zero jobs reaches positive infinity meanwhile the increment ε reaches positive zero. Then, we can get that

$$\begin{array}{rcl} \frac{C(\phi)}{C(\psi)} & = & \lim\limits_{n\to+\infty} \lim\limits_{\varepsilon\to 0^+} \frac{(n+1)(t+1)}{(n+1)(r_{j+1}+\varepsilon)+1} \\ & \geqslant & \lim\limits_{n\to+\infty} \lim\limits_{\varepsilon\to 0^+} \frac{(r_j+1)}{(r_{j+1}+\varepsilon)+\frac{1}{n+1}} \geqslant \frac{r_j+1}{r_{j+1}} \\ & = & \frac{1}{r_2} \end{array} \qquad (1)$$

2. If no job is scheduled at time $t \in [r_j, r_{j+1})$ for any $j \in \mathbb{N}$, it means that the first scheduled job will be processed at $t \in [r_n, +\infty)$. Suppose there are n unit processing time jobs. We have that

$$\begin{array}{rcl} \frac{C(\phi)}{C(\psi)} & = & \frac{nt+\frac{n(n+1)}{2}}{\frac{n(n+1)}{2}} = 1 + \frac{2t}{n+1} \\ & \geq & 1 + \frac{2r_n}{n+1} \\ & = & 1 + \frac{2}{n+1} \cdot \frac{r_2(r_2^{n-1}-1)}{r_2-1}. \end{array} \qquad (2)$$

According to the definition of ρ, it must be $\rho \geq \max\{(1), (2)\}$, which achieves its maximum when (1) equals to (2). Considering the discreteness of n, we solve this equation by a numerical way. Thus, we obtain the solution of $n = 4$ and $r_2 \approx 0.6499$. Therefore, we show that one lower bound of competitive ratio ρ of any semi-online algorithm $\mathbb{A}$ with r_t^1 equals to $1/r_2 \approx 1.5387$. □

4.2 Upper Bound Analysis

Due to the NP-hardness of this problem in [11], it is almost impossible to attain an optimal schedule without global information. We introduce the optimal preemptive schedule ψ to substitute with optimal schedule π. Let $C(\varphi)$ define the total completion time of schedule φ, $C(\pi)$ define that of schedule π and $C(\psi)$ define that of schedule ψ. It is easy to know $\frac{C(\varphi)}{C(\pi)} \leq \frac{C(\varphi)}{C(\psi)}$ and thus $\rho \leq \max\limits_{I\in\mathfrak{C}} \left\{\frac{C(\varphi)}{C(\psi)}\right\}$. Furthermore, any schedule is dependent on the information of the given instance I. Actually, $\varphi = \varphi(I)$, $\pi = \pi(I)$ and $\psi = \psi(I)$.

Suppose the starting time of the block is denoted as v_1. It is certain that there is some job j_0 such that $v_1 = r_{j_0}$. In the following discussion, we show that any instance in $\mathfrak{C}$ can be concluded into a kind of instances which satisfy $v_1 = r_n$. Let *Block* define the contiguously processing block. Obviously, *Block* is combined with three kinds of jobs S_- ,S and S_+, ie. $Block = S_- \bigcup S \bigcup S_+$ where

$$\begin{array}{rcl} S_- & = & \{j | r_j < v_1, j \in Block\}, \\ S & = & \{j | r_j = v_1, j \in Block\}, \\ S_+ & = & \{j | r_j > v_1, j \in Block\}. \end{array}$$

For ease of exposition, we define S_+^0 as jobs with zero processing time[2] in S_+. Similarly, S^0 represents jobs with zero processing time in S. Apparently, S_- does not contain any jobs with zero processing time, otherwise it will break the rules of algorithm AOPT. Let $Block^0$ define the jobs with zero processing time in any instance, ie. $Block^0 = S^0 \bigcup S_+^0$.

Lemma 2. *For any instance $I \in \mathfrak{C}$, if $|S^0| \geq 2$ there must be a job $j \in Block^0$. Let $I' = I - \{j\}$. We have that $\frac{C(\varphi)}{C(\psi)} \leq \max\left\{\frac{C(\varphi')}{C(\psi')}, 1+\alpha\right\}$.*

Proof. Since $Block^0 = S^0 \bigcup S_+^0$, a job $j \in Block^0$ means that either $j \in S^0$ or $j \in S_+^0$. We will show that under any circumstances the inequality holds.

$j \in S^0$: Since $|S^0| \geq 2$, the removal of job $j \in S^0$ will not dismiss the time $r_j = r_t^1$ during the time period between r_{j-1} and r_j. That is to say the new schedule φ' will not change regardless of job j. Therefore, it holds that $\frac{C(\varphi)}{C(\psi)} = \frac{C(\varphi')+r_j}{C(\psi')+r_j} \leq \max\left\{\frac{C(\varphi')}{C(\psi')}, 1\right\}$ for $j \in S^0$.

$j \in S_+^0$: Since $p_j = 0$, there must be an 'immediate predecessor' job $\kappa \in Block$ such that $C_\kappa = s_j + 0 = C_j$. According to AOPT, $C_\kappa = s_\kappa + p_\kappa \leq (1+\alpha)r_{s_\kappa}^1$ and $r_{s_\kappa}^1 \leq r_j$. Thus, we get that $\frac{C_j}{r_j} \leq \frac{(1+\alpha)r_j}{r_j} = 1+\alpha$. Therefore, $\frac{C(\varphi)}{C(\psi)} \leq \max\left\{\frac{C(\varphi')}{C(\psi')}, 1+\alpha\right\}$ for $j \in S_+^0$.

In summary, $\frac{C(\varphi)}{C(\psi)} \leq \max\left\{\frac{C(\varphi')}{C(\psi')}, 1+\alpha\right\}$ and φ' does not change regardless of job j under any circumstances. □

After recursively applying lemma 2 to a timely updated instance I, we could get an instance I' in which either $S = \{j | p_j = 0, r_j = v_1\}$ where $|S| = 1$ or $S =$

[2]We allow $r_{j+1} = r_j$. Otherwise, there is no zero processing time job.

$\{j|p_j > 0, r_j = v_1\}$ where $|S| \geq 1$. Moreover, $|S_+^0| = 0$ means S_+ has no zero processing time jobs.

Without loss of generality, suppose $S_+ \neq \varnothing$. Define r_{max} as the latest release time of all jobs, ie. $r_{max} = \max\{r_j\} = \max\{r_j|j \in S_+\} > v_1$. Let $S_+^{end} \triangleq \{j \in S_+|r_j = r_{max}\}$, which means jobs in S_+ with latest release time. For r_{max}, there is a job κ such that $s_\kappa < r_{max} \leq C_\kappa$. Classify S_+^{end} with two kinds of jobs S_+^{long} and S_+^{short}, where $S_+^{long} = \{j \in S_+^{end}|p_j \geq p_\kappa\}$ and $S_+^{short} = \{j \in S_+^{end}|p_j < p_\kappa\}$. It is obvious that $S_+^{short} \bigcup S_+^{long} = S_+^{end} \subset S_+$.

Lemma 3. *For any instance $I \in \mathfrak{C}$, if there is a job $j \in S_+^{long}$, reduce r_j to r_κ and let I' be the instance after reduction. We get that $\frac{C(\varphi)}{C(\psi)} \leq \frac{C(\varphi')}{C(\psi')}$.*

Proof. According to the definition of S_+^{long}, it stands that $p_j \geq p_\kappa$ for all $j \in S_+^{long}$. Thus $t + p_j > (1+\alpha)r_t^1$ for time period $r_\kappa \leq t < s_\kappa$. It illustrates that if job j reduces its release time r_j to r_κ it will never be scheduled during $r_\kappa \leq t < s_\kappa$. Job κ also satisfies $s_\kappa + p_\kappa \leq (1+\alpha)r_{s_\kappa}^1$ which means job κ is the job with minimal processing time in Ω_{s_κ}. Hence, it holds that $\varphi = \varphi'$ which could deduce $C(\varphi) = C(\varphi')$.

Since the reduction of release time will not increase the total completion time $C(\psi)$ of the optimal preemptive schedule ψ. We have $C(\psi) \geq C(\psi')$.

Therefore, it holds that $\frac{C(\varphi)}{C(\psi)} \leq \frac{C(\varphi')}{C(\psi')}$. □

Lemma 4. *Given that $I \in \mathfrak{C}$ whose $S_+^{long} = \varnothing$, for any job $j \in S_+$ change the processing time of j from p_j to $p_j + \delta$, where $\delta \in \{\delta^-, \delta^+\}$, $\delta^- = -\min\{p_j|j \in S_+^{end}\}$ and $\delta^+ = p_\kappa - \max\{p_j|j \in S_+^{end}\}$. Let $\varphi(\delta)$ and $\psi(\delta)$ be the objective value of AOPT and optimal preemptive algorithm respectively. Then we obtain that $\frac{C(\varphi)}{C(\psi)} \leq \max\left\{\frac{C(\varphi(\delta^-))}{C(\psi(\delta^-))}, \frac{C(\varphi(\delta^+))}{C(\psi(\delta^+))}\right\}$.*

Proof. Since $S_+^{long} = \varnothing$, we know that $0 \leq p_j + \delta \leq p_\kappa$ for all $j \in S_+^{end}$. Moreover, the schedule of jobs after the latest release time r_{max} will remain the same due to the same increments of the processing time. Therefore, $\varphi = \varphi(\delta)$ and the objective value $C(\varphi(\delta))$ of schedule $\varphi(\delta)$ is a linear function with variant δ.

With regards to the optimal preemptive schedule ψ, the number of jobs being preempted by j will increase (decrease) with the decrease (increase) of the processing time of job j, which means the objective value of schedule ψ is a piecewise linear concave function with variant δ.

According to proposition (1) in appendix, we have $\frac{C(\varphi)}{C(\psi)} \leq \max\left\{\frac{C(\varphi(\delta^-))}{C(\psi(\delta^-))}, \frac{C(\varphi(\delta^+))}{C(\psi(\delta^+))}\right\}$. □

Define $I_1 \in \mathfrak{C}$ as an arbitrary instance whose $S_+ = \varnothing$ and $|S| \geq 2$ with $p_j > 0$ for all $j \in S$. Rearrange the indexes of jobs in S by $\{j_q(S)\}_{q=1}^{|S|}$. Let Sub_k be the k-th contiguously processing sub-block in S illustrated in Figure 1, ie. $Sub_k = \{j_m(S), ..., j_{m+t}(S)|m \leq q \leq m+t-1,\ C_{j_q(S)} = s_{j_{q+1}(S)}\}$. Suppose that there are totally K sub-blocks of Sub_k. Denote $s(Sub_k) = s_{j_m(S)} = \min\{s_j|j \in Sub_k\}$ as the earliest starting time of sub-block Sub_q for $k = 1, ..., K$.

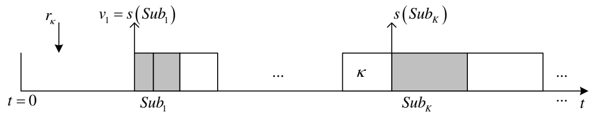

Figure 1: The structure of set S

For the instance I_1, it can be concluded from $S_+ = \varnothing$ that $v_1 = r_{max}$. The reduction of release time from $r_{j_q(S)}$ to r_κ has no influence on the original schedule φ. Since the processing time remains, we know that $C(\varphi) = C(\varphi')$. Similar to the proof of lemma 4, the objective value of ψ' will not increase since the release time is reduced, which means $C(\psi) \geq C(\psi')$. Hence, we have lemma 5.

Lemma 5. *For any instance $I_1 \in \mathfrak{C}$, if there exists a job $\kappa \in S_-$ with $C_\kappa = s(Sub_K)$ then reduce the release time of all jobs in Sub_K from v_1 to r_κ $(r_\kappa < v_1)$. Let I_1' be the new instance. We have $\frac{C(\varphi)}{C(\psi)} \leq \frac{C(\varphi')}{C(\psi')}$.*

In the following content, we only discuss instances whose S_+ is an empty set and S owns one sub-block namely $Sub_1(S)$ in which all processing time is positive. Here we define a class of instances as I_2, which must satisfy $I_2 \in \mathfrak{C}$ and $I_2 = S_- \bigcup S$ where $S = Sub_1(S)$.

Similar to lemma 4, we give lemma 6 without proof.

Lemma 6. *For any instance $I_2 \in \mathfrak{C}$, if $|S| \geq 2$ define the immediate successor job after S by $j_a = \{j \in S_-|s_j = C(Sub_1)\}$. Let $\delta^- = -\min\{p_j|j \in S^0\}$ and $\delta^+ = p_a - \max\{p_j|j \in S^0\}$. For any $j \in S^0$ set $p_j = p_j + \delta$ where $\delta \in \{\delta^-, \delta^+\}$. Let I_2' be the new instance after such modification. We have $\frac{C(\varphi)}{C(\psi)} \leq \max\left\{\frac{C(\varphi(\delta^-))}{C(\psi(\delta^-))}, \frac{C(\varphi(\delta^+))}{C(\psi(\delta^+))}\right\}$.*

Here we reformulate the description of one kind of simple instances. It is easy to prove that the simpler instance is still in $\mathfrak{C}$. Let I denote the newly simplified instance with properties as follows. $I = S_- \bigcup S$ with N jobs and release time $r_j \leq r_{j+1} (j = 1, ..., n-1)$. Moreover, $|S| = 1$ means only one job, no matter it is a zero processing time job scheduled at $t = v_1$ or non-zero processing time job scheduled at $t \geq v_1$.

Define $p = \min\{p_j|p_j > 0, j \in I\}$ as the non-zero minimal processing time jobs in I. Let r_e be the earli-

est release time of jobs with processing time p. Denote
$p^{(2)} = \min\{p_j | j \in I, p_j > p\}$,
$u = \min\limits_{e \leqslant i \leqslant n-1} \left\{ \frac{(1+\alpha)^{(i+1-e)}}{\alpha}(r_{i+1} - r_i) \right\}$ *and*
$l = -\min\limits_{e \leqslant i \leqslant n-1} \left\{ (1+\alpha)^{(i+1-e)}(r_{i+1} - r_i) \right\}$.
We construct an instance I' with the following methods. During each transformation,

```
1.  Find out the $p$-kind jobs and $r_e$.
2.  Let $p = p + \Delta$. For all jobs with $r_j > r_e$,
    let $r_j = r_j + \Delta r_j$, where
    $\Delta r_j = \Delta \cdot (1+\alpha)^{e-j}$ for $\Delta > 0$, and
    $\Delta r_j = \frac{\Delta}{\alpha} \cdot \left(1 - (1+\alpha)^{e-j}\right)$ for $\Delta < 0$.
3.  Update instance $I'$.
```

Since the schedule will not be changed after changing process, we have that $C(\varphi)$ is a linear function with the variant Δ.

With the increase of index i, $\{\Delta r_i\}$ is a series of monotonously decreasing sequence which approaches positively to zero. It means the number of preempting jobs will increase. Thus the function of $C(\psi)$ is a piecewise linear and concave function with Δ.

Similarly, while $\Delta = \Delta^-$ we can also get $C(\varphi)$ is a linear function and $C(\psi)$ is a piecewise linear and concave function with Δ.

Therefore, we get $\frac{C(\varphi)}{C(\pi)} \le \max\left\{\frac{C(\varphi(\Delta^-))}{C(\psi(\Delta^-))}, \frac{C(\varphi(\Delta^+))}{C(\psi(\Delta^+))}\right\}$ due to proposition 1 in appendix.

Lemma 7. *For any instance $I \in \mathfrak{C}$, let $I' = I'(\Delta)$ be the 'simpler' instance transformed from I by above techniques where $\Delta \in \{\Delta^-, \Delta^+\}$, $\Delta^- = \max\{-p, l\}$ and $\Delta^+ = \min\{p^{(2)} - p, u\}$. We have $\frac{C(\varphi)}{C(\psi)} \le \max\left\{\frac{C(\varphi(\Delta^-))}{C(\psi(\Delta^-))}, \frac{C(\varphi(\Delta^+))}{C(\psi(\Delta^+))}\right\}$.*

Theorem 3 (Upper Bound). *For any instance $I \in \mathfrak{C}$, the upper bound of its performance ratio ρ is $\frac{157}{100} = 1.57$.*

Proof. After such modification from lemma (1) to lemma (7), an arbitrary instance in $\mathfrak{C}$ should finally be transformed into three cases: (Suppose there are totally n jobs in *Block*)

Case 1: *Block* $= S$ and $|S| = 1$.
Suppose the processing time of the only job in *Block* is p_s, then we have $\frac{C(\varphi)}{C(\psi)} = \frac{v_1 + p_s}{v_1 + p_s} = 1$. Actually, it is impossible to have only one job according to our assumption of the next release time.

Case 2: *Blcok* $= S_- \bigcup S$, $p_j = p(j \in S_-)$, $p_k = 0(k \in S)$ and $|S| = 1$.
In this case,

$$\frac{C(\phi)}{C(\psi)} = \frac{n \cdot v_1 + \frac{n(n-1)}{2} \cdot}{v_1 + \frac{n(n-1)}{2} \cdot p} = 1 + \frac{n-1}{1 + \frac{n(n-1)}{2} \cdot \frac{p}{v_1}}. \quad (3)$$

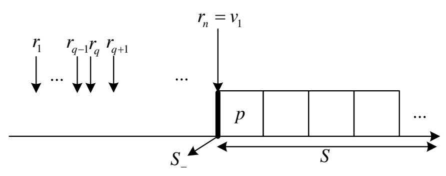

Figure 2: The instance structure of Case 2.

Since for all $j \in \{1, ..., n-1\}$ the inequality that $r_j + p > (1+\alpha)r_{j+1}$ holds, we could get that

$$\frac{p}{v_1} > \frac{\alpha}{1 - (1+\alpha)^{1-n}} - \frac{r_1}{r_n} \cdot \frac{\alpha}{1 - (1+\alpha)^{1-n}} \quad (4)$$

According to (3), (4) and $r_1 \ge 0$, we know

$$\frac{C(\phi)}{C(\psi)} \le 1 + \frac{n-1}{1 + \frac{n(n-1)}{2} \cdot \frac{\alpha}{1-(1+\alpha)^{1-n}}} \quad (5)$$

Let $f(n, \alpha) = \frac{1}{n-1} + \frac{\alpha}{2} \cdot \frac{n}{1-(1+\alpha)^{(1-n)}}$. It is easy to know that $\alpha = \arg\min_\alpha \left\{\max_n \frac{C(\varphi)}{C(\psi)}\right\} = \arg\min_\alpha \{\min_n f(n, \alpha)\}$. Since we have

$$\begin{aligned} f'(n, \alpha) &= \frac{d}{dn} f(n, \alpha) \\ &= \frac{\alpha}{2} \left\{ \frac{1}{1-(1+\alpha)^{1-n}} - \frac{n(1+\alpha)^{1-n}\ln(1+\alpha)}{1-(1+\alpha)^{1-n}} \right\} \\ &\quad + \frac{-1}{(n-1)^2} \end{aligned} \quad (6)$$

for $\alpha \in [0.5, 1)$ and
$n = 2, 3 \quad : \quad f_n'(n, \alpha) < 0, \forall \alpha \in [0.5, 1)$;
$n \ge 4 \quad : \quad f_n'(n, \alpha) > 0, \forall \alpha \in [0.5, 1)$,
we have $\min_n f(n, \alpha) = \min\{f(3, \alpha), f(4, \alpha)\}$.

Therefore, there exists a unique $\alpha_1 \approx 0.7338$ and for any $\alpha \in [0.5, \alpha_1)$ the 'best' number of jobs should be $n = 4$. Similarly, the 'best' number of jobs should be chosen as $n = 3$ when $\alpha \in [\alpha_1, 1)$.

Case 3: *Blcok* $= S_- \bigcup S$, $p_j = p(j \in S_-)$, $p_k = p(k \in S)$ and $|S| = 1$.

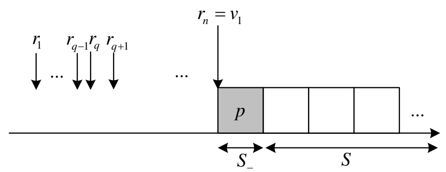

Figure 3: The instance structure of Case 3.

Here we utilize the same method as that of Case 2. We have that for $\alpha \in [0.5, \alpha_2)$ the 'best' number of jobs should be $n = 4$ while the 'best' number of jobs should be chosen as $n = 3$ for $\alpha \in [\alpha_2, 1)$.

With the result of lemma 2 and the analysis of above three cases, we have that for a fixed value of $\alpha \in [0.5, 1)$

$$\begin{aligned}
\frac{C(\varphi)}{C(\psi)} &\leqslant \max_{n}\left\{1+\alpha, 1+\frac{1}{f(n,\alpha)}, 1+\frac{2}{g(n,\alpha)}\right\} \\
&\leqslant 1+\begin{cases} \max\limits_{n=4}\left\{\alpha, \frac{1}{f(n,\alpha)}, \frac{2}{g(n,\alpha)}\right\}, \alpha \in [0.5, \alpha_2) \\ \max\limits_{n}\left\{\alpha, \frac{1}{f(4,\alpha)}, \frac{2}{g(3,\alpha)}\right\}, \alpha \in [\alpha_2, \alpha_1) \\ \max\limits_{n=3}\left\{\alpha, \frac{1}{f(n,\alpha)}, \frac{2}{g(n,\alpha)}\right\}, \alpha \in [\alpha_1, 1) \end{cases} \\
&= 1+\max_{n=4}\left\{\alpha, \frac{1}{f(n,\alpha)}, \frac{2}{g(n,\alpha)}\right\}, \alpha \in [0.5, \alpha_2) \\
&= 1+\frac{1}{f(4,0.5)} = \frac{157}{100} = 1.57.
\end{aligned} \tag{7}$$

Therefore, we have proved that

$$\begin{aligned}
\alpha^* &= \arg\min_{\alpha}\left\{\max_{n\geq 2}\frac{C(\varphi)}{C(\psi)}\right\} = 0.5, \\
n^* &= \arg\max_{n\geq 2}\left\{\frac{C(\varphi)}{C(\psi)}\right\} = 4 \quad (\alpha = 0.5), \\
\rho &\leq \frac{C(\varphi^*)}{C(\psi^*)} = \min_{\alpha}\max_{n}\frac{C(\varphi)}{C(\psi)} = \frac{157}{100} = 1.57.
\end{aligned}$$

□

According to theorems 2 and 3, we have the theorem of the competitive ratio as follows.

Theorem 4 (**Competitive Ratio**). *For any instance $I \in \mathfrak{C}$, the competitive ratio ρ of algorithm AOPT satisfies $\rho \in [1.5387, 1.57]$ with a narrow gap of 0.0313, which presents algorithm AOPT is an almost-optimal semi-online algorithm for the problem $1|r_j, r_t^1, \mathfrak{C}|\sum C_j$.*

5 Conclusion

Our semi-online model of $1|r_j, r_t^1, \mathfrak{C}|\sum C_j$ covers a wide range of scheduling models in a kind of single processor models, including periodic, aperiodic and sporadic models. No matter these models are schedulable or not, sustainable or not, our model of $1|r_j, r_t^1, \mathfrak{C}|\sum C_j$ contains them. Of course, in a practical view, schedulable or sustainable models seem to be much more useful than others. Thus the instance-space of such models must be the subset of $\mathfrak{C}$, which is easy to conclude that the competitive ratio of such ́feasible' models by AOPT must be less than 1.57. Furthermore, our algorithm AOPT is easy for judging, executing and responding in a real time environment. Additionally, we show the power of introduction of some future information (r_t^1), which greatly improves the competitive ratio from best online result of 2 to our semi-online result of $[1.5387, 1.57]$.

References

[1] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems.* Prentice Hall, 2002.

[2] JA Hoogeveen and A.P.A. Vestjens. Optimal On-Line Algorithms for Single-Machine Scheduling. *Lecture Notes in Computer Science*, pages 404–414, 1996.

[3] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 520–529. ACM New York, NY, USA, 2001.

[4] J.A. Hoogeveen and A.P.A. Vestjens. A Best Possible Deterministic On-line Algorithm for Minimize Maximum Delivery Time on a Single Machine. *SIAM Journal on Discrete Mathematics*, pages 56–63, 2000.

[5] F. Li, J. Sethuraman, and C. Stein. An optimal online algorithm for packet scheduling with agreeable deadlines. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 801–802. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2005.

[6] Y. Bartal, F.Y.L. Chin, M. Chrobak, S.P.Y. Fung, W. Jawor, R. Lavi, J. Sgall, and T. Tichy. Online competitive algorithms for maximizing weighted throughput of unit jobs. In *Proceedings of the 21st Symposium on Theoretical Aspects of Computer Science*, pages 187–198. Springer, 2004.

[7] Junjie Zhou, Ye Tao, Zhijun Chao, Yugeng Xi. Competitive Analysis of a New Semi-online Algorithm to Minimize Total Completion Time with Next Release Time on a Single Machine. *under review, Elsevier Editorial System(tm) for Discrete Applied Mathematics*, 2008.

[8] F.Y.L. Chin and S.P.Y. Fung. Online scheduling with partial job values: Does timesharing or randomization help? *Algorithmica*, 37(3):149–164, 2003.

[9] S. Seiden, J. Sgall, and G. Woeginger. Semi-online scheduling with decreasing job sizes. *Operations Research Letters*, 27(5):215–221, 2000.

[10] W.E. Smith. Various Optimizers for Single-Stage Production. 1955.

[11] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.R. Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5(2):287–326, 1979.

Appendix

Proposition 1. *Let $f(x)$ and $g(x)$ be two positive functions defined on the interval $[x_1, x_2] \in \mathbb{R}^1$, respectively. If $f(x)$ is a convex function while $g(x)$ is a concave function, then $\frac{f(x)}{g(x)}$ reaches its maximum at either endpoint of the interval: ie. $\frac{f(x)}{g(x)} \leq \max\left\{\frac{f(x_1)}{g(x_1)}, \frac{f(x_2)}{g(x_2)}\right\}, \quad \forall x \in [x_1, x_2]$.*