Journal of Artificial Intelligence Research 56 (2016) 429-461 Submitted 02/16; published 07/16

Efficient Mechanism Design for Online Scheduling

Xujin Chen XCHEN@AMSS.AC.CN
Xiaodong Hu XDHU@AMSS.AC.CN
AMSS, Chinese Academy of Science, Beijing, China

Tie-Yan Liu TYLIU@MICROSOFT.COM
Weidong Ma WEIMA@MICROSOFT.COM
Tao Qin TAOQIN@MICROSOFT.COM
Microsoft Research, Beijing, China

Pingzhong Tang KENSHIN@MAIL.TSINGHUA.EDU.CN
Tsinghua University, Beijing, China

Changjun Wang WCJ@AMSS.AC.CN
Beijing University of Technology, Beijing, China

Bo Zheng ZHENGB10@MAILS.TSINGHUA.EDU.CN
Tsinghua University, Beijing, China

Abstract

This paper concerns the mechanism design for online scheduling in a strategic setting. In this setting, each job is owned by a self-interested agent who may misreport the release time, deadline, length, and value of her job, while we need to determine not only the schedule of the jobs, but also the payment of each agent. We focus on the design of incentive compatible (IC) mechanisms, and study the maximization of social welfare (i.e., the aggregated value of completed jobs) by competitive analysis. We first derive two lower bounds on the competitive ratio of any deterministic IC mechanism to characterize the landscape of our research: one bound is 5, which holds for equal-length jobs; the other bound is $\frac{\kappa}{\ln \kappa} + 1 - o(1)$, which holds for unequal-length jobs, where κ is the maximum ratio between lengths of any two jobs. We then propose a deterministic IC mechanism and show that such a simple mechanism works very well for two models: (1) In the preemption-restart model, the mechanism can achieve the optimal competitive ratio of 5 for equal-length jobs and a near optimal ratio of $(\frac{1}{(1-\epsilon)^2} + o(1))\frac{\kappa}{\ln \kappa}$ for unequal-length jobs, where $0 < \epsilon < 1$ is a small constant; (2) In the preemption-resume model, the mechanism can achieve the optimal competitive ratio of 5 for equal-length jobs and a near optimal competitive ratio (within factor 2) for unequal-length jobs.

1. Introduction

Online scheduling has been widely studied in the literature (Baruah, Koren, Mao, Mishra, Raghunathan, Rosier, Shasha, & Wang, 1992; Baruah, Haritsa, & Sharma, 1994; Porter, 2004; Zheng, Fung, Chan, Chin, Poon, & Wong, 2006; Ting, 2008), where each job is characterized by a release time, a deadline, a length, and a value for its successful completion by the deadline. Inspired by emerging areas like computational economics and cloud computing, we consider a strategic setting of the online scheduling problem, where each job is owned by a self-interested agent and she may have the incentive to manipulate the scheduling algorithm in order to be better off. To be specific, the agent may deliberately delay the release time of her job, inflate its length, and misreport its value and deadline.

Given this situation, a carefully designed online scheduling mechanism is needed to regulate the strategic behaviors of the agents and to (approximately) optimize some system objectives. In this work, we focus on the maximization of social welfare, i.e., the total value of completed jobs.[1] We use *competitive analysis* (Lavi & Nisan, 2004) to evaluate the performance of such a mechanism, which compares the social welfare implemented by the mechanism (without any knowledge of all future jobs) with that of the optimal offline allocation (with the knowledge of future jobs).

In this work, we consider two scheduling models: the *preemption-restart* model (Ting, 2008) and the *preemption-resume* model (Porter, 2004). Once preempted, jobs in the first model have to restart from the beginning; while jobs in the second model can resume from the break point. Since preemption is always assumed in this work, the two models are also referred to as *restart* model and *resume* model, respectively, and their involved jobs are called *non-resumable* and *resumable*, respectively.

1.1 Problem Formulation

We consider online scheduling models with infinite time period $T = \mathbb{R}_{\geq 0}$. Suppose there is a single machine that processes at most one job at any given time. Jobs come over time, and we use J to denote the set of jobs. Each job $j \in J$ is owned by a self-interested agent (which is also denoted as j for simplicity); and it is characterized by a private type $\theta_j = (r_j, d_j, l_j, v_j) \in T \times T \times \mathbb{R}_{>0} \times \mathbb{R}_{>0}$, where r_j is the release time[2], d_j is the deadline, l_j is the length (i.e., the processing time), and v_j is the value if the job is completed by its deadline.

A resumable job j is *completed* if and only if it is processed for l_j time units in total between its release time r_j and deadline d_j, while a non-resumable job j is *completed* if and only if it is processed for l_j consecutive time units between its release time r_j and deadline d_j.

Let $\kappa = \max_{i,j \in J} \frac{l_i}{l_j}$ be the maximum ratio between the lengths of any two jobs. For simplicity, we assume all job lengths are normalized, i.e., $l_j \in [1, \kappa]$ for all $j \in J$, and assume κ is known in advance following the practice in the work of Chan et al. (2004) and Ting (2008).

We study direct revelation mechanisms, in which each agent participates by simply declaring the type of her job $\hat{\theta}_j = (\hat{r}_j, \hat{d}_j, \hat{l}_j, \hat{v}_j)$ at time $\hat{r}_j$. We use $\hat{\theta}$ to denote the profile of reported types of all the agents. Given the declared types of the agents, a mechanism M is used to schedule/allocate the jobs and determine the payment of each agent. Here we only consider "reasonable" mechanisms which (1) do not schedule a job after its reported deadline and (2) do not schedule a job once it has been processed for a reported length.

Given a certain mechanism M and a job sequence $\hat{\theta}$, we use $q_j(\hat{\theta}, t)$ to denote whether job j is completed by time t (if it is completed, $q_j(\hat{\theta}, t) = 1$; otherwise $q_j(\hat{\theta}, t) = 0$). Then

1. It is also referred as weighted *throughput* in the scheduling literature.
2. Note that release time is also referred as *arrival time* in the online auction literature (Parkes, 2007). It is the earliest time at which the agent has full knowledge of her job. Thus it is the earliest time the job is available to the scheduling process.

the value that agent j extracts from the mechanism can be represented by $q_j(\hat{\theta}, d_j)v_j$, and the social welfare of the mechanism can be represented by $W(M, \theta) = \sum_j q_j(\hat{\theta}, d_j)v_j$.

Let $p_j(\hat{\theta})$ denote the amount of money that the mechanism charges agent j. We assume that agents have quasi-linear preferences (Nisan, 2007), i.e., the utility of agent j is $u_j(\hat{\theta}, \theta_j) = q_j(\hat{\theta}, d_j)v_j - p_j(\hat{\theta})$.

Since agents are self-interested, they may misreport their types in a strategic way. It is easy to see that the misreport of a shorter length is a dominated strategy; otherwise, her job cannot be completed even if it is scheduled by the mechanism (since $\hat{l}_j < l_j$). Therefore, the agents will not underreport the lengths of their jobs. Similar to the work of Porter (2004), we assume that the system will not return a completed job to agent j until $\hat{d}_j$.[3] In this way, we restrict the agent's report to be $\hat{d}_j \leq d_j$. In addition, we assume that no agent has knowledge about her job before its release time, so we also have $\hat{r}_j \geq r_j$.

Considering the potential misreport of the agents, we are concerned with incentive compatible and individually rational mechanisms. A mechanism is *incentive compatible* (IC) if, for any agent j, regardless of the behaviors of other agents, truthful reporting her own type maximizes her utility. A mechanism is *individually rational* (IR) if for each job j, truthful reporting leads to a non-negative utility. In addition, we would also like the mechanism to (approximately) maximize social welfare. We say a mechanism M is (strictly) *c-competitive* if there does not exist any job sequence θ such that $c \cdot W(M, \theta) < W(opt, \theta)$, where *opt* denotes the optimal offline mechanism[4]. Sometimes we also say that M has a competitive ratio of c.

1.2 Related Work

The online scheduling problem has been studied in both the non-strategic setting (Lipton & Tomkins, 1994; Borodin & El-Yaniv, 1998; Bar-Noy, Guha, Naor, & Schieber, 2001; Zheng et al., 2006; Kolen, Lenstra, Papadimitriou, & Spieksma, 2007; Ting, 2008; Nguyen, 2011) (whose focus is algorithm design) and the strategic setting (Nisan & Ronen, 2001; Lavi & Nisan, 2004; Friedman & Parkes, 2003; Porter, 2004; Hajiaghayi, Kleinberg, Mahdian, & Parkes, 2005; Parkes, 2007) (whose focus is mechanism design).

Non-strategic setting. For the case of $\kappa = 1$, a lower bound of 4 on the competitive ratio of any deterministic algorithm is given by Woeginger (1994). A 4.56-competitive deterministic algorithm is constructed by Zheng et al. (2006) for the restart model, and a 4.24-competitive deterministic algorithm is designed by Kim (2011) for both restart and resume models. A 2-competitive randomized algorithm is introduced for restart model in the work of Fung et al. (2014), and a lower bound of 1.693 is provided in the work of Epstein and Levin (2010). By restricting release time and deadlines to be integers, a randomized algorithm with competitive ratio $\frac{e}{e-1} \approx 1.582$ is proposed by Chin et al. (2006), and a deterministic algorithm with competitive ratio $2\sqrt{2} - 1 \approx 1.828$ is proposed by Englert et

3. Actually, it should be viewed as a decision by the mechanism designer rather than an "assumption". This decision is crucial to ensure the incentive compatibility which we will see later.
4. Since we only care about the social welfare performance of *opt* in the competitive analysis, which only depends on the schedule, regardless of the payments, we also call *opt* as "optimal offline allocation", or simply "optimal allocation".

al. (2012). The best lower bounds currently are 1.25 for randomized algorithms (Chin & Fung, 2003) and 1.618 for deterministic algorithms (Hajek, 2001).

For general values of κ, a lower bound of $\sqrt{\kappa}$ on the competitive ratio of any deterministic algorithm is derived in the work of Chan et al. (2004). The lower bound is improved to $\frac{\kappa}{2\ln\kappa} - 1$ by Ting and Fung (2008), and an algorithm with competitive ratio $\frac{6\kappa}{\log\kappa} + O(\kappa^{5/6})$ is given for the restart model. The scheduling problem with discrete time is considered in the work of Durr, Jez and Nguyen (2012). In particular, the lower bound is improved to $\frac{\kappa}{\ln\kappa} - o(1)$, and a $(3 + o(1))\frac{\kappa}{\ln\kappa}$-competitive algorithm is designed for the resume model. A randomized algorithm with competitive ratio $O(\log(\kappa))$ and a lower bound of $\Omega(\sqrt{\frac{\log\kappa}{\log\log\kappa}})$ is provided by Canetti and Irani (1998).

Assuming the maximum ratio between the value densities (value divided by length) of any two jobs is bounded above by a known number ρ, a $(1+\sqrt{\rho})^2$-competitive algorithm is given by Koren and Shasha (1995). The bound $(1+\sqrt{\rho})^2$ is optimal as a matching lower bound is given by Baruah et al. (1992).

There is also a rich literature concerned with non-preemptive scheduling (Lipton & Tomkins, 1994; Goldman, Parwatikar, & Suri, 2000; Goldwasser, 2003; Ding & Zhang, 2006; Ding, Ebenlendr, Sgall, & Zhang, 2007; Ebenlendr & Sgall, 2009). However, it can be easily verified that an algorithm with bounded competitive ratio cannot be designed in the setting of unrestricted values and arbitrary release time. Therefore, the most common assumption added in the non-preemptive scheduling problem is *proportional values*, i.e., the value of each job is proportional to the length. In the work of Goldman et al. (2000), a tight upper and lower bound of 2 are given for the deterministic competitiveness when all jobs have equal length (thus, equal value), and a $6(\lfloor\log_2\kappa\rfloor + 1)$-competitive randomized algorithm is provided for general value of κ, matching the $\Omega(\log\kappa)$ lower bound (Lipton & Tomkins, 1994) within a constant factor.

Strategic setting. In the work of Lavi and Nisan (2015), by assuming integer time points, a scheduling problem for the $\kappa = 1$ case is studied. The authors show that there is no incentive compatible mechanism which can obtain a constant competitive ratio, if the payment must be made when the job is completed. Hence, they propose a family of "semi-myopic" algorithms with competitive ratio 3, under the assumption of semi-myopic strategies. In the work of Hajiaghayi et al. (2005), a specific scheduling problem in which $\kappa = 1$ is considered under the restart model. A deterministic IC mechanism with competitive ratio 5 is designed, and a lower bound of 2 is given to any deterministic IC mechanism. However, to our knowledge, the case $\kappa > 1$ in either the restart model or the resume model has not been studied from the perspective of mechanism design (considering the incentive issues). Our work fills this gap.

Assuming the maximum ratio between the value densities (value divided by length) of any two jobs is bounded above by a known number ρ, an IC mechanism with a competitive ratio of $(1+\sqrt{\rho})^2 + 1$ is designed by Porter (2004), and it is proved that $(1+\sqrt{\rho})^2 + 1$ is a lower bound of the competitive ratio for any deterministic mechanism.

Recently, online scheduling mechanisms have been investigated in cloud computing (Zaman & Grosu, 2012; Azar, Ben-Aroya, Devanur, & Jain, 2013; Zhang, Li, Jiang, Liu, Vasilakos, & Liu, 2013; Lucier, Menache, Naor, & Yaniv, 2013; Mashayekhy, Nejad, Grosu, & Vasilakos, 2014; Wu, Gu, Li, Tao, Chen, & Ma, 2014). In these works, mechanisms are designed to allocate computational resources to users, and users can use those virtual machines during the entire period requested. In these model, jobs are non-preemptive, which differs from our setting.

1.3 Our Results

Our main results can be summarized as follows.

First, in order to characterize the boundary of our research, we derive two lower bounds on the competitive ratio for any online deterministic IC mechanism. One bound is 5, which holds for the situation where all the jobs have equal length (i.e., $\kappa = 1$). This bound improves the previous lower bound of 2 (Hajiaghayi et al., 2005). The other bound is $\frac{\kappa}{\ln \kappa} + 1 - o(1)$, which characterizes the asymptotical property of the competitive ratio when the variance of job lengths, i.e., κ, is sufficiently large.

Second, we design a simple mechanism Γ_1 and prove that in both the restart and resume models Γ_1 is not only IC, but also achieves good social welfare.

- In the restart model, Γ_1 has a competitive ratio of $\kappa + 2 + (1 + \frac{1}{\kappa})^{\kappa}$ when κ is small (in particular, the ratio is 5 for $\kappa = 1$), and $(\frac{1}{(1-\epsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln \kappa}$ when κ is large ($\kappa \geq 16$ is enough), where $0 < \epsilon < 1$ is a small constant.
- In the resume model, Γ_1 has a competitive ratio of $(\kappa + 1)(1 + \frac{1}{\kappa})^{\kappa} + 1$ when κ is small (in particular, the ratio is 5 for $\kappa = 1$), and $(\frac{2}{(1-\epsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln \kappa}$ when κ is large ($\kappa \geq 16$ is enough), which is just slightly worse than that for the restart model (within a factor of 2).

It is also worth mentioning that:

- Comparing with the lower bounds, we can see that, in both the restart and resume models, Γ_1 is optimal for equal-length jobs ($\kappa = 1$), and near optimal (within a constant factor) for unequal-length jobs.
- In comparison with the best-known algorithms without considering incentive compatibility, asymptotically speaking, Γ_1 improves the best-known ratio $\frac{6\kappa}{\log \kappa} + O(\kappa^{\frac{5}{6}})$ (Ting, 2008) in the restart model to $(\frac{1}{(1-\epsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln \kappa}$; and improves the best-known ratio $(3 + o(1)) \cdot \frac{\kappa}{\ln \kappa}$ (Dürr, Jeż, & Nguyen, 2012) in the resume model to $(\frac{2}{(1-\epsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln \kappa}$. Thus even if one does not care about the strategic aspect, Γ_1 would still be a very nice algorithm to use.

Note that designing mechanisms for online scheduling problems is generally difficult since it combines the challenges of mechanism design (i.e., ensuring incentive compatibility) with the challenges of online algorithms (i.e., dealing with uncertainty about future inputs). We would like to highlight the main techniques used in this work to tackle these challenges.

(1) The allocation rule of mechanism Γ_1 uses a carefully selected function to trade-off three key elements: value, length, and degree of completion. The trade-off function is delicate in the sense that it ensures both the efficiency and the monotonicity which is crucial to the incentive compatibility.

(2) In order to obtain good competitive ratios for the resume model, we design two non-trivial virtual charging schemes to bound the performance of the proposed mechanism: the *integral charging scheme* and the *segmental charging scheme*.

While we focus on single machine model in this paper, our work extends to multiple identical machines. One way of the extension is similar to the work of Lucier et al. (2013), in which it is assumed that at most h machines can be allocated to each job at any given time, and the parameter h stands for a common parallelism bound of the system. The details of this extension can be found in Appendix E. Another way to extend our results to multiple identical machines is to assume that each job j needs a fixed number of machines when it is processed. Please refer to our working paper (Ma, Zheng, Qin, Tang, & Liu, 2014) for more details.[5]

2. Lower Bounds

In this section, we present two lower bounds on the competitive ratio of any deterministic IC mechanism, which hold for both the restart and resume models.

The competitive analysis can be interpreted as a game between the designer of the online mechanism and an adversary. Given mechanism Γ_1, the adversary selects the sequence of jobs that maximizes the competitive ratio, the ratio of the social welfare obtained by an offline optimal algorithm over the social welfare obtained by Γ_1. Therefore, the key of proving lower bounds is to construct subtle adversary behaviors.

We first introduce two notions, the *dominant job* and the *shadow job*.

Definition 2.1 (**Dominant Job**). *For a deterministic IC mechanism with competitive ratio c, job i is called a* dominant job *at its release time r_i, if and only if v_i is larger than c times the total value of all other jobs whose release time is no later than r_i.*

It is easy to see that, in order to obtain a reasonable competitive ratio, if a dominant job i has a tight deadline, then the mechanism must schedule i at its release time r_i. Otherwise, consider the case in which no more jobs are released after r_i. In this case, the mechanism cannot obtain a competitive ratio of c if it gives up the dominant job i.

Definition 2.2 (**Shadow Job**). *Suppose a job i has a tight deadline, i.e., $d_i = r_i + l_i$, then job i' is called a* shadow job *of i, if i' has the same parameters (r_i, l_i, v_i) as i, except for a later deadline ($d'_i > d_i$).*

Clearly, the shadow job i' is more flexible and can be completed later. As for *shadow jobs*, we show that the following lemma holds for any IC mechanism with a non-trivial competitive ratio.

Lemma 2.3 (**Shadow Job Argument**). *For a deterministic IC mechanism Γ with a non-trivial competitive ratio c, if Γ completes a job i (with tight deadline d_i) under some scenario I, then under scenario I', which substitutes some shadow job i' for job i, Γ must also complete job i' at time d_i.*

5. In the working paper, we only consider the restart model, and ignore the misreport of release time or deadline.

Proof. Suppose Γ has not completed job i' at d_i under scenario I', we could consider a subsidiary scenario I'', which includes all jobs in scenario I' and adds on several *dominant jobs*. Remember that we call some job *dominant* if its value is sufficiently large (see Definition 2.1). These dominant jobs are released one by one at $d_i, d_i + 1, \ldots, d_i + \lfloor d'_i - d_i \rfloor$ respectively, and denoted as $0, 1, \ldots, \lfloor d'_i - d_i \rfloor$ accordingly, where d_i is the deadline of job i and d'_i is the deadline of shadow job i'. What's more, each of these dominant jobs is of unit length and has a tight deadline. We claim that, to achieve the desired (non-trivial) competitive ratio, Γ must complete all these dominant jobs, thus the time interval $[d_i, d'_i)$ is occupied. (The reason is as follows: if Γ does not schedule any dominant job $j \in \{0, 1, \ldots, \lfloor d'_i - d_i \rfloor\}$, then we consider a scenario I''', which only includes jobs with release time no later than $d_i + j$ in I''. Since scenario I''' is indistinguishable from I'' up to time $d_i + j$, we know Γ does not schedule the dominant job j in scenario I''', hence cannot obtain a competitive ratio of c.)

Because the subsidiary scenario I'' is indistinguishable from scenario I' up to time d_i, job i' will not be completed at d_i. Furthermore, because of the existence of dominant jobs, job i' will not be completed finally. However, if job i' falsely declares its type to be the same as that of job i, i.e., misreports its deadline to be d_i, it would be completed at time d_i and be better off, contradicting the incentive compatibility[6]. $\square$

In the following, we will derive lower bounds leveraging Lemma 2.3. First, the following theorem specifies a lower bound when jobs have equal length (i.e., $\kappa = 1$). Note that our result concerns the strategic setting, while Woeginger (1994) shows that the competitive ratio of any deterministic algorithm in the non-strategic setting is at least 4.

Theorem 2.4. *When $\kappa = 1$, no deterministic IC mechanism can obtain a competitive ratio less than* 5.

To prove the theorem, in addition to using an adversary argument similar to that in the work of Woeginger (1994), we need to further perturb the job sequence and leverage the shadow job argument.

Intuitively, we construct a special job set, in which tight-deadline jobs are released one by one, and any two jobs collide with each other (that is, the deadline of one job is later than the release time of the other, and under any mechanism, it is impossible for these two jobs to be both completed). The values of these jobs are carefully selected such that a later released job is more valuable than the earlier one (predecessor), and the value difference between such two neighboring jobs is constrained by a small-enough additive constant. Furthermore, in such a job set, the values of the first and last jobs are set to obey a specific amplification. Along with the execution of any mechanism, the adversary would release a series of such job sets. Once the mechanism completes one job, the adversary stops releasing any job. The subtleness lies in choosing the time to release such job sets: once the mechanism almost completes some job a in a job set, the adversary may release a new job set whose jobs all collide with job a but do not collide with the predecessor of job a. In this way, if the mechanism would not abandon the current job a but complete it, then there should be an optimal allocation which completes: (1) several jobs in the previous job sets, (2) the most

6. The above scenario contradicts the monotonicity condition (see a strict definition at start of Section 3.2); And Theorem 1.15 of the work of Parkes (2007) shows that monotonicity is necessary for incentive compatibility.

valuable job (i.e., the last job) in the newly released job set, and (3) the job a.[7] However, the mechanism can only complete job a. This discrepancy leads to the lower bound of competitive ratio. The detailed proof can be found below.

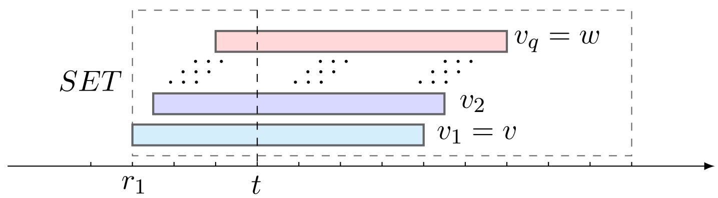

Figure 1: Structure of $SET(v, w, t, \delta)$

Proof. Suppose by contradiction that there exists a deterministic IC mechanism Γ which achieves a competitive ratio of $5 - \epsilon$ for some $0 < \epsilon < 1$. We adopt the notation of SET introduced by Woeginger (1994). Define $SET(v, w, t, \delta)$ (for $w \geq v > 0$, $t > 0$ and $\delta > 0$) as a set of jobs $\{1, 2, \ldots, q\}$ satisfying the following properties:

(1) $v_1 = v$, $v_q = w$, and $v_j < v_{j+1} \leq v_j + \delta$ for $1 \leq j \leq q-1$. Hence, q can be any integer no less than $\lceil \frac{w-v}{\delta} \rceil$. We call δ as the *magnifying parameter* of a SET.

(2) $l_j = d_j - r_j = 1, \forall j$, i.e., all jobs are unit-length and have tight deadlines.

(3) $0 \leq r_1 < \cdots < r_q < t < d_1 < \cdots < d_q$, thus, any two jobs collide with each other. We call t as the *split point* of a SET.

We define the release time of a SET as the release time of its first job. Figure 1 shows the visual structure of $SET(v, w, t, \delta)$. The adversary behavior is as follows.

Adversary Behavior: The adversary will release some SETs one after another depending on Γ. First, $SET_0 = SET(1, \alpha, 1/2, \delta)$ is released at time 0, where $\alpha = 4 - \epsilon/2$ and $\delta < \epsilon/4$. From the definition of SET, we know that the first job in SET_0 has value 1, the last job in SET_0 has value α, and the value difference between any two neighboring jobs is upper bounded by δ.

Next, we specify: (1) when will the adversary release a new $SET_i (i \geq 1)$, and (2) how the adversary sets the parameters of $SET_i (i \geq 1)$. For (1), we specify by Algorithm 1. The notations used in Algorithm 1 are detailed in Table 1.

7. In the proof, we construct a new scenario, in which job a is perturbed to have later deadline, thus can be completed later. We make use of the shadow job argument in the analysis, which makes the lower bound increased by 1, compared with the previous lower bound in the non-strategic setting.

Table 1: Summary of notation in the proof of Theorem 2.4

SET_i	i-th released SET, in full, $SET(v_{i1}, w_i, t_i, \delta_i)$
job ij	j-th job in SET_i.
r_{ij}, d_{ij} and v_{ij}	release time, deadline and value of job ij.
w_i	value of the last job in SET_i
t_i	split point of SET_i
δ_i	magnifying parameter of SET_i
job i^*	trigger job in SET_{i-1}
job $\hat{i}$	the preceding job of i^* in SET_{i-1}

Algorithm 1: The Adversary Behavior

1: **Initial:** Release SET_0 at time 0.
2: **while** Γ has not completed any job, **do**
3:   **if** Γ *almost* completes the j-th job ($j \geq 2$) in SET_i (Precisely, Γ has been executing job ij for $d_{i(j-1)} - r_{ij}$ period of time since r_{ij}). **then**
4:     Release SET_{i+1} at time $d_{i(j-1)}$.
5:   **else**
6:     Do not release any other job.
7:   **end if**
8: **end while**

It is worth mentioning that: (i) SET_{i+1} is only triggered when a non-first job in SET_i is almost completed, and we call such a job a *trigger job*. (ii) No more SET will be released once some job is completed by Γ.

Suppose the *trigger jobs* in $SET_0, \ldots, SET_{i-1}$ are named $1^*, \ldots, i^*$ successively. Accordingly, we denote the job with release time just earlier than each trigger job as $\hat{1}, \ldots, \hat{i}$, and we call them *preceding jobs*. From Line 4 of Algorithm 1, we know that each new SET_i is released at the deadline of $\hat{i}$. Note that trigger job i^* and its preceding job $\hat{i}$ are both located in SET_{i-1}.

We now specify the parameters of $SET_i = SET(v_{i1}, w_i, t_i, \delta_i)$, $i \geq 1$. Remember that SET_0 is defined as $SET(1, \alpha, 1/2, \delta)$, in which $\alpha = 4 - \epsilon/2$ and $\delta < \epsilon/4$.

- The adversary sets v_{i1} equal to the value of the trigger job i^* in SET_{i-1}, that is $v_{i^*} = v_{i1}$. Note that v_{i1} is the value of the first job in SET_i.
- The adversary sets $\delta_i = \delta/2^i$, $w_i = \max\{(\alpha - 1)v_{i1} - \sum_{j=1}^{i-1} v_{j1}, v_{i1}\}$ for $i \geq 2$, and $w_1 = (\alpha - 1)v_{11}$.
- The adversary sets $t_i = (d_{i^*} + d_{\hat{i}})/2$, where d_{i^*} and $d_{\hat{i}}$ are deadlines of trigger job i^* and its preceding job $\hat{i}$. Note that by setting $t_i = (d_{i^*} + d_{\hat{i}})/2$, all jobs in SET_i are released after $d_{\hat{i}}$ but before d_{i^*}. Hence, all the new jobs collide with trigger job i^* and none of them collides with job $\hat{i}$.

Figure 2 illustrates how the adversary releases a new SET by an example. In this example, Γ almost completes the j-th job ($j \geq 2$) in SET_i. SET_{i+1} is released at the deadline of job $i(j-1)$, and the value of the first job of SET_{i+1} is equal to v_{ij}.

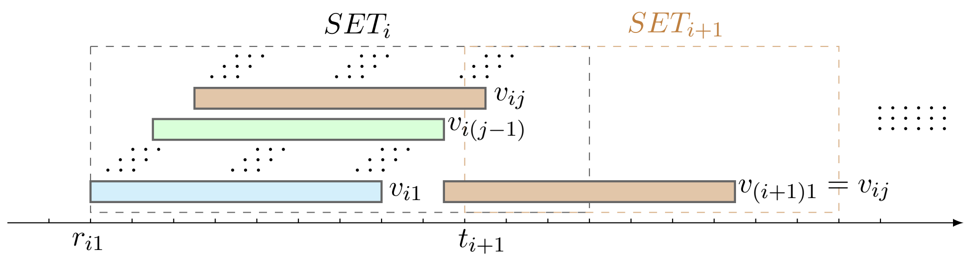

Figure 2: An example of SET_{i+1} and SET_i

According to Algorithm 1, if Γ always gives up trigger jobs and switches to schedule some job in the newly released SET, the adversary will release new SETs one after another. One may wonder whether the adversary will release new SETs infinitely. In other words, will subscript of SET_i tend to infinity?

The answer is no, which can be seen from the definition of w_i. Since $2 < \alpha < 4$, by Lemma 4.3 of the work of Woeginger (1994), after a finite number (denote k) of steps, v_{k1} must be no less than the corresponding sum term $(\alpha - 1)v_{k1} - \sum_{j=1}^{k-1} v_{j1}$, and $w_k = v_{k1}$ must hold. Remember that v_{k1} and w_k denote the value of first job and last job in SET_k respectively, thus there exists only one job in SET_k. According to Algorithm 1, no matter whether Γ completes this job or not, the adversary will not release any other job. Therefore, SET_k is the ultimate SET and job $k1$ is the ultimate job.

So far, we have clarified the adversary's behaviors. Next, we show how to derive the lower bound based on such an adversary.

According to Algorithm 1 and the structure of SET, we know the adversary allows Γ to complete at most one job. Actually, the completed job can be: (1) the first job in SET_0 (i.e., job 01); (2) a trigger job i^*, $1 \le i \le k$; or the first job in SET_i, $1 \le i < k$ (i.e., job $i1$); or (3) the ultimate job $k1$. Let us analyze them one by one.

(1) If Γ completes job 01, then we consider a scenario in which job 01 is substituted by its shadow job $01'$, whose deadline is late enough (i.e., even if it started being executed from the deadline of last job in SET_0, it still can be completed in time). According to Lemma 2.3, mechanism Γ must complete job $01'$ at time 1, and thus abandon the last job (with value $w_0 = \alpha$) in SET_0. Therefore, it only obtains a social welfare of $v_{01} = 1$. However, the optimal allocation (which first completes the last job in SET_0 and then job $01'$) obtains a social welfare of $\alpha + 1$. This contradicts the fact that Γ has a competitive ratio of $5 - \epsilon$, since $\alpha + 1 = (4 - \epsilon/2) + 1 > 5 - \epsilon$.

(2) If Γ completes a trigger job i^* or a job $i1$, $1 \le i \le k$, without loss of generality, we denote such job as job ij, and we know $v_{ij} = v_{i^*} = v_{i1}$. If Γ completes job ij, $1 \le i \le k$, then similarly, we consider a scenario in which job ij is substituted by its shadow job $(ij)'$, whose deadline is late enough. By Lemma 2.3, Γ must complete job $(ij)'$ at time d_{ij}, obtaining a social welfare of $v_{ij} = v_{i^*} = v_{i1}$. However, social welfare of the optimal allocation (which completes jobs $\hat{1}, \ldots, \hat{i}$, the last job in SET_i, and then job $(ij)'$) is at least $\sum_{j=1}^{i} v_{\hat{j}} + w_i + v_{ij} > \sum_{j=1}^{i}(v_{j1} - \delta_{j-1}) + w_i + v_{ij} >$

$\sum_{j=1}^{i} v_{j1} - 2\delta + (\alpha - 1)v_{i1} - \sum_{j=1}^{i-1} v_{j1} + v_{ij} > (\alpha + 1)v_{i1} - \epsilon/2 > (5 - \epsilon)v_{i1}$. This contradicts the fact that Γ has a competitive ratio of $5 - \epsilon$.

(3) If Γ completes the ultimate job $k1$, we consider a scenario in which the adversary releases two copies of job $k1$ in SET_k. Clearly, in this scenario, Γ will choose one copy to complete. We denote the completed copy as job $(k1)_1$ and the other as job $(k1)_2$. We then consider a scenario in which job $(k1)_1$ is substituted by its shadow job $(k1)'$, whose deadline is unit-time later than that of job $k1$. According to Lemma 2.3, Γ must complete job $(k1)'$ at d_{k1} and obtains a social welfare of v_{k1}. However, the optimal allocation (which completes jobs $\hat{1}, \ldots, \hat{k}$, job $(k1)_2$, and then job $(k1)'$) can obtain a social welfare of at least $\sum_{j=1}^{k} v_{\hat{j}} + v_{k1} + v_{k1} > \sum_{j=1}^{k}(v_{j1} - \delta_{j-1}) + w_k + v_{k1} > \sum_{j=1}^{k} v_j - 2\delta + (\alpha - 1)v_{k1} - \sum_{j=1}^{k-1} v_{j1} + v_{k1} > (\alpha + 1)v_{k1} - \epsilon/2 > (5 - \epsilon)v_{k1}$. Remember that in SET_k, we have $v_{k1} = w_k = (\alpha - 1)v_{k1} - \sum_{j=1}^{k-1} v_{j1}$. This contradicts the fact that Γ has a competitive ratio of $5 - \epsilon$.

□

Second, to understand the asymptotic property of the lower bound when κ is large, we construct scenarios inspired by the example of Durr et al. (2012) and obtain the following theorem.

Theorem 2.5. *When κ is sufficiently large, no deterministic IC mechanism can obtain a competitive ratio less than $\frac{\kappa}{\ln \kappa} + 1 - o(1)$. In particular, no deterministic IC mechanism can obtain a competitive ratio less than $\frac{\kappa}{\ln \kappa} + 0.94$ for $\kappa \geq 16$.*

Proof. For convenience of analysis, we denote $\alpha = \frac{\kappa}{\ln \kappa}$, $r = \lceil \alpha \rceil - 1$, and assume $\kappa \geq 16$. Let us consider the following adversary behaviors.

Adversary Behavior: At time 0, a long job B with type $\theta_B = (0, \kappa, \kappa, \alpha)$ is released, as well as two short jobs a_1 and $\dot{a}_1$ with the same type $(0, 1, 1, 1)$. Moreover, at each integer moment $0 \leq t \leq \kappa - 1$, if the mechanism schedules only job B in $[0, t)$, then two short jobs a_{t+1} and $\dot{a}_{t+1}$ of unit length are released at t, with tight deadline $t + 1$, and no new job is released otherwise. The values of jobs a_t and $\dot{a}_t$ satisfy:

$$v(a_t) = v(\dot{a}_t) = \begin{cases} 1 & \text{if } t < \alpha, \\ e^{\frac{t}{\alpha} - 1} & \text{if } t \geq \alpha. \end{cases} \tag{1}$$

Note that job a_t and job $\dot{a}_t$ are of the same type, and the cases analyzed below for a_{t_0} can be naturally applied to $\dot{a}_{t_0}$.

According to the adversary behavior, we know the adversary allows Γ to complete at most one job. Actually, the completed job can be: (1) a job a_{t_0} with $t_0 < \alpha$; (2) a job a_{t_0} with $t_0 \geq \alpha$; or (3) job B. We analyze these three cases as follows.

(1) If the mechanism schedules a job a_{t_0} with $t_0 < \alpha$, then we consider a scenario that includes jobs $B, a_1, \dot{a}_1, \ldots, a_{t_0-1}, \dot{a}_{t_0-1}, \dot{a}_{t_0}$ and job a'_{t_0}. Here, job a'_{t_0} with type $(t_0 - 1, \kappa + 1, 1, 1)$, is a shadow job of a_{t_0}. According to Lemma 2.3, the mechanism must complete job a'_{t_0} at t_0 and only obtains a social welfare of 1. However, in this scenario, the optimal mechanism will complete job B first, and then schedule a'_{t_0} at time κ and complete it, with the optimal social welfare $\alpha + 1$. So the ratio is $\alpha + 1$.

(2) If the mechanism schedules a job a_{t_0} with $t_0 \geq \alpha$, then we consider a scenario that includes jobs $B, a_1, \dot{a}_1, \ldots, a_{t_0-1}, \dot{a}_{t_0-1}, \dot{a}_{t_0}$, and job a'_{t_0}. Here, job a'_{t_0} with type $(t_0 - 1, t_0 + 1, 1, e^{t_0/\alpha - 1})$, is a shadow job of a_{t_0}. According to Lemma 2.3, the mechanism should schedule job a'_{t_0} at time t_0 and complete only a'_{t_0}. Thus, the mechanism only obtains a social welfare of $v(a'_{t_0})$. However, one of the optimal mechanisms will schedule and complete all jobs $\dot{a}_t$ for $t = 1, \ldots, t_0$, and then schedule a'_{t_0} at time t_0 and complete it, resulting in the following optimal social welfare

$$\begin{aligned}
&\lceil\alpha\rceil - 1 + \sum_{t=\lceil\alpha\rceil}^{t_0} e^{\frac{t}{\alpha}-1} + e^{\frac{t_0}{\alpha}-1} = r + \sum_{t=r+1}^{t_0} e^{\frac{t}{\alpha}-1} + e^{\frac{t_0}{\alpha}-1} \geq r + \int_r^{t_0} e^{\frac{t}{\alpha}-1} + e^{\frac{t_0}{\alpha}-1} \\
=& r - \alpha e^{\frac{r}{\alpha}-1} + (\alpha+1)e^{\frac{t_0}{\alpha}-1} = f(\alpha, r) + (\alpha+1)e^{\frac{t_0}{\alpha}-1} = f(\alpha, r) + (\alpha+1)v(a'_{t_0}).
\end{aligned}$$

Here, we have introduced a function f defined as $f(\alpha, r) \equiv r - \alpha e^{\frac{r}{\alpha}-1}$. Considering $\alpha = \frac{\kappa}{\ln \kappa}$ and $r = \alpha - 1$, we have $\alpha - r \in (0, 1]$. As $e^x \geq 1 + x$ and both sides converge to 1 as x approaches 0, we have

$$f(\alpha, r) = r - \alpha e^{\frac{r}{\alpha}-1} \leq r - \alpha \cdot \frac{r}{\alpha} = 0, \tag{2}$$

and $f(\alpha, r)$ approaches 0 as κ grows. So the ratio is $\alpha + 1 - o(1)$.

(3) If the mechanism schedules and completes job B, obtaining a social welfare of α, then we consider a scenario that includes jobs $a_1, \dot{a}_1, \ldots, a_\kappa, \dot{a}_\kappa$ and job B'. Here, job B' with type $\theta_{B'} = (0, 2\kappa, \kappa, \alpha)$, is a shadow job of B. Similarly, we claim that an IC mechanism should schedule job B' at time 0 and complete it at time κ. Thus, the mechanism only obtains a social welfare of $v(B')$. However, one of the optimal mechanisms will schedule and complete all κ small jobs a_t from $t = 0$ to $\kappa - 1$, and then schedule and complete job B'. This leads to a social welfare at least

$$\begin{aligned}
&\lceil\alpha\rceil - 1 + \sum_{t=\lceil\alpha\rceil}^{\kappa} e^{\frac{t}{\alpha}-1} + \alpha = r + \sum_{t=r+1}^{\kappa} e^{\frac{t}{\alpha}-1} + \alpha \geq r + \int_r^{\kappa} e^{\frac{t}{\alpha}-1} + \alpha \\
=& r - \alpha e^{\frac{r}{\alpha}-1} + \alpha e^{\frac{\kappa}{\alpha}-1} + \alpha = f(\alpha, r) + \alpha e^{\frac{\kappa}{\alpha}-1} + \alpha = f(\alpha, r) + \alpha e^{\ln\kappa - 1} + \alpha \\
=& f(\alpha, r) + \alpha \cdot \frac{\kappa}{e} + \alpha = f(\alpha, r) + \alpha^2 \cdot \frac{\ln\kappa}{e} + \alpha = f(\alpha, r) + (\alpha \cdot \frac{\ln\kappa}{e} + 1)v(B').
\end{aligned} \tag{3}$$

When $\kappa \geq 16$, we have $e \leq \ln\kappa$. Then the above equation is larger than $f(\alpha, r) + (\alpha + 1)v(B')$. Therefore the ratio is $\alpha + 1 - o(1)$.

Combining the three cases together, we prove the nonexistence of $(\frac{\kappa}{\ln\kappa} + 1 - o(1))$-competitive mechanisms. Since $f(\alpha, r) \geq -0.06$ when $\kappa \geq 16$, the competitive ratio is at least $\frac{\kappa}{\ln\kappa} + 0.94$ for $\kappa \geq 16$. □

3. Mechanism Design

In this section, we describe a simple mechanism Γ_1 (whose allocation and payment rules are given in Algorithm 2), which works surprisingly well for both the restart and resume models, and handles the settings with different values of κ in a unified framework. In contrast, previous works (Dürr et al., 2012) need to design separate and very different algorithms to deal with different values of κ.

3.1 The Mechanism Γ_1

Before introducing our mechanism, we first introduce the concept of the *valid active time* of an uncompleted job j, until time t, denoted as

$$e_j(t) := \begin{cases} t - \min\{s | x(t') = j, \forall t' \in [s, t)\}, & \text{for the } \textit{restart} \text{ model} \\ \int_0^t \mu(x(s) = j) ds, & \text{for the } \textit{resume} \text{ model} \end{cases} \tag{4}$$

where $x(t)$ is the mechanism's allocation function, which maps each time point to an available job, or to 0 if the machine is idle.[8] And $\mu(\cdot)$ is an indicator function that returns 1 if the argument is true, and zero otherwise. Note that $e_j(\cdot)$ can also take a vector θ as an argument. For example, $e_j(\theta, t)$ is shorthand for the $e_j(t)$ for the job sequence θ.

It can be seen that in the restart model, at time t, if a job j has received an allocation at time $t' < t$ and has not been preempted after that, then $e_j(t) = t - t'$. In the resume model, $e_j(t)$ is the accumulated processing time of job j until time t.

We say that a job j is *feasible* at time t if (1) its reported release time is before t; (2) it has not been completed yet; and (3) it has enough time to be completed before its reported deadline, i.e., $\hat{d}_j - t \geq \hat{l}_j - e_j(t)$. We use $J_F(t)$ to denote the set of all feasible jobs at time t.

According to Algorithm 2, at any time t, Γ_1 assigns a priority score, $\hat{v}_j \cdot \beta^{\hat{l}_j - e_j(\hat{\theta}, t)}$, to each feasible job $j \in J_F(t)$, and always processes the feasible job with the highest priority (ties are broken in favor of the job with the smaller $\hat{r}_j$). Here β is located in $(0, 1)$ and will be determined later during the competitive analysis. The payment rule of Γ_1 is essentially the critical-value payment (Parkes, 2007), which is similar to that of the second-price auction. Hence, the payment is equal to the minimum bid the agents have to make to remain allocated.[9] In the following pseudocode, $\hat{\theta}_{-j}$ denotes the reported types of all jobs other than j.

8. In Equation 4, since s=t is a valid candidate for the minimization, if there does not exist an s, s.t., $x(t') = j, \forall t' \in [s, t)$ in the restart model, then $e_j(t) = 0$.
9. Note that we use the critical-value payment, so the payment of a completed job j depends on other jobs' types between $\hat{r_j}$ and $\hat{d_j}$. If our mechanism allows returning completed job before its reported deadline, the calculation of critical-value payment will face trouble: it is possible that agent j misreports a much later deadline to obtain a cheaper payment, but his job is completed and returned before its true deadline. That is the reason why we restrict our mechanism to return completed job at its reported deadline. It is worth mentioning that, if the payment must be made when the job is completed, (Lavi & Nisan, 2015) has shown that there is no incentive compatible mechanism which can obtain a constant competitive ratio.

Algorithm 2:

Allocation Rule
for all time t **do**
  if $J_F(t) \neq \emptyset$ **then**
    $x(t) \leftarrow \arg\max_{j \in J_F(t)}(\hat{v}_j \cdot \beta^{\hat{l}_j - e_j(\hat{\theta},t)})$
  else $x(t) \leftarrow 0$
end
Payment Rule
for all job j **do**
  if $q_j(\hat{\theta}, \hat{d}_j) = 1$ **then**
    $p_j(\hat{\theta}) = \min(v'_j | q_j(((\hat{r}_j, \hat{d}_j, \hat{l}_j, v'_j), \hat{\theta}_{-j}), \hat{d}_j) = 1)$
  else $p_j(\hat{\theta}) = 0$
end

The intuition of our mechanism is two-fold. First, to ensure efficiency, one must trade value against length: a job with a larger value has a higher priority, and a job with a larger remaining length has a lower priority. Γ_1 uses a simple priority function to achieve the tradeoff: as can be seen, the priority score $\hat{v}_j \cdot \beta^{\hat{l}_j - e_j(\hat{\theta},t)}$ of a job is positively correlated with its value and negatively correlated with its remaining length. Second, to ensure IC, Γ_1 uses the critical-value payment rule and a monotone[10] allocation rule.

Note that both the allocation rule and the payment rule can be implemented efficiently. For the allocation rule, it is enough to consider the time point when some new jobs arrive or some existing jobs are completed. And, we give algorithms in Appendix A to show that the payment for each agent can be computed in polynomial time.

Clearly, because of the critical-value payment rule, Γ_1 is individually rational. In the following subsection, we prove its incentive compatibility.

3.2 Incentive Compatibility

We call an allocation rule of a mechanism *monotone*, if a job with truthfully reported type $\theta_j = (r_j, d_j, l_j, v_j)$ cannot be completed in the mechanism, then a dominated[11] declaration of its type $\hat{\theta}_j = (\hat{r}_j, \hat{d}_j, \hat{l}_j, \hat{v}_j)$ cannot make it completed either.

According to Theorem 1.13 of the work of Parkes (2007), in order to establish the truthfulness of a mechanism, it is enough to prove the monotonicity of its allocation rule.

Theorem 3.1. *Mechanism Γ_1 is incentive compatible, in both the restart model and resume model.*

Proof. We prove the monotonicity of the allocation rule of Γ_1. Assume a job j is not completed under Γ_1 when θ_j is truthfully declared (we denote this case as $True$). We now show that j cannot be completed either by declaring $\hat{\theta}_j = (\hat{r}_j, \hat{d}_j, \hat{l}_j, \hat{v}_j)$, where $\hat{r}_j \geq r_j$, $\hat{l}_j \geq l_j$, $\hat{d}_j \leq d_j$ and $\hat{v}_j \leq v_j$. And we denote any such case as $False$.

Suppose job j has ever been executed for $k > 0$ times in the $True$ case, we define the following points in the execution of job j: let t_i^s and t_i^p be the ith time that job j starts

10. We have a strict definition of monotonicity at start of Section 3.2.
11. We say a type $\hat{\theta}_j$ is dominated by type θ_j (denoted as $\hat{\theta}_j \prec \theta_j$) if $\hat{r}_j \geq r_j$, $\hat{d}_j \leq d_j$, $\hat{l}_j \geq l_j$ and $\hat{v}_j \leq v_j$.

execution and is preempted respectively, where $i = 1, 2, \ldots, k$, and let $t^a = \arg\inf_t(e_j(t) + \hat{d}_j - t < \hat{l}_j)$ be the time that job j is abandoned. If job j is never started, then we set $t^s_1 = t^p_1 = t^a$.

We also refer to $P = [r_j, t^s_1) \cup [t^p_1, t^s_2) \ldots \cup [t^p_k, t^a] = P_0 \cup P_1 \ldots \cup P_k$ as the *pending period* of job j, and $A = [t^s_1, t^p_1) \cup [t^s_2, t^p_2) \ldots \cup [t^s_k, t^p_k) = A_1 \cup A_2 \ldots \cup A_k$ as the *executing period* of job j.

We first consider monotonicity with regard to $\hat{r}_j$, regardless of other variables. Clearly, from the definition of t^a, declaring $\hat{r}_j > t^a$ could not cause the job to be completed. Thus, we can restrict our attention to $\hat{r}_j \in [r_j, t^a] = P \cup A$.

A necessary condition for job j to be completed (in $False$) is that job j should be executed sometime in the period P. However, according to Lemma 3.2 (see below), job j cannot be executed in P. Therefore, declaring $\hat{r}_j \geq r_j$ cannot cause the job to be completed.

Intuitively, Lemma 3.2 says that, under case $True$ and $False$, the set of jobs that are scheduled in the period P must be the same. Thus, job j cannot be executed in period P.

We then consider $\hat{d}_j$, $\hat{l}_j$ and $\hat{v}_j$. The proof is essentially the same as the proof of $\hat{r}_j$: declaring $\hat{d}_j \leq d_j$, $\hat{l}_j \geq l_j$ and $\hat{v}_j \leq v_j$ will not improve job j's priority, and as a result, there cannot be a change in the execution of jobs in the pending period P. So declaring $\hat{d}_j \leq d_j$, $\hat{l}_j \geq l_j$ and $\hat{v}_j \leq v_j$ cannot cause the job to be completed. This proves that the allocation rule of Γ_1 is monotone. □

In the following, we formally introduce Lemma 3.2, which is used in the above theorem. For this lemma we introduce some additional notation: under case $True$ and $False$, denote by $\mathcal{J}$ and $\hat{\mathcal{J}}$ respectively the set of jobs which have ever been executed in P, and denote by $\mathcal{I}$ and $\hat{\mathcal{I}}$ respectively the set of jobs which have ever been pending in A.

Lemma 3.2. *(1) $\mathcal{I} \cap \mathcal{J} = \emptyset$, (2) $\mathcal{I} \cap \hat{\mathcal{J}} = \emptyset$, (3) $\mathcal{J} = \hat{\mathcal{J}}$.*

Proof. Consider a job $i \in \mathcal{I}$, according to the defintion of $\mathcal{I}$, under case $True$, job i has lower priority than job j in period $A \cup P$.

Relation (1) means that, under case $True$, job i cannot be executed in period P. It is obvious, since job j (with higher priority than i) is pending in period P.

Relation (2) means that, under case $False$, job i cannot be executed in period P either. We prove this by contradiction. Suppose job i is executed at some time point in P. We denote $t_i = \min\{t \in P | x(t) = i\}$, and assume $t_i \in P_n$ for $0 \leq n \leq k$. We have an observation for the pending period P_n, $0 \leq n \leq k$.

Observation 3.3. *In pending period P_n, if Γ_1 schedules jobs by a sequence*[12] *of $j^1_n \ldots j^{h(n)}_n$ ($h(n) \geq 1$, is the number of such active jobs in P_n) under case $True$, then we know (1) the release time of each job $j^2_n \ldots j^{h(n)}_n$ is in the period P_n; in particular, the release time of job j^1_n in P_n ($n \geq 1$) is exactly time t^p_n (2) each job $j^1_n \ldots j^{h(n)}_n$ is either completed or abandoned in P_n; and there is no idle time in P_n.*

Here, we use $f_j(t)$ to denote the priority of job j at time t. Suppose that, under case $True$, it is job j^i_n (one of $j^1_n \ldots j^{h(n)}_n$) that is executed at t_i, and its priority is $f_{j^i_n}(t_i)$. Then

12. A job may appear more than once in the sequence if it is preempted and resumed/restarted later.

under case $False$, since i is executed at t_i, according to Observation 3.3, we can deduce that the priority of job i at time t_i, i.e., $f_i(t_i)$ must be larger than $f_{j_n^i}(t_i)$.

Therefore, we can deduce that i must have been executed sometime in the period $U_i = (A_1 \cup \ldots \cup A_n)$. Otherwise, i should also be executed at time t_i under case $True$, contradicting the fact that $i \in \mathcal{I}$. Similarly, we denote $s_i = \min\{t \in U_i | x(t) = i\}$, and assume $s_i \in A_m$ for $1 \leq m \leq n$.

We claim, under case $False$, the priority of job i at time s_i, i.e., $f_i(s_i)$ satisfies the inequality as below.

$$f_i(s_i) > \begin{cases} f_{j_n^i}(t_i) \cdot \beta^{|A_n|+|A_{n-1}|+\cdots+|A_{m+1}|+|t_m^p - s_i|}, & \text{if } m \leq n-1; \\ f_{j_n^i}(t_i) \cdot \beta^{|t_m^p - s_i|}, & \text{if } m = n. \end{cases}$$

Otherwise, the priority of job i at time t_i is at most $f_{j_n^i}(t_i)$ (consider the case that all the periods $[s_i, t_m^p), A_{m+1}, \ldots, A_{n-1}, A_n$ are allocated to i).

According to the definition of s_i, we know s_i is the first time that i is executed in period A. Therefore, the priority of job i at s_i remains the same when shifting from case $True$ to case $False$. However, under case $True$, job j is executed at time s_i (hence, with a priority larger than $f_i(s_i)$), and all the periods $[s_i, t_m^p), A_{m+1}, \ldots, A_{n-1}, A_n$ are allocated to j. Therefore, at time t_i, job j will have a priority larger than $f_{j_n^i}(t_i)$, contradicting the fact that j_n^i is executed at time t^i.

Relation (3) means that, no matter case $True$ or case $False$, the jobs that are executed in the period P are the same. Relation (3) can be derived naturally from Relation (2). □

4. Competitive Analysis

In this section, we show that mechanism Γ_1 performs quite well in terms of social welfare by comparison with the optimal offline allocation, which has full knowledge of the future jobs at the beginning of the execution.

To perform the competitive analysis, we need to design *virtual charging schemes*. Under a certain virtual charging scheme, for every job j completed by the optimal allocation *opt*, we charge its value (or partial value) to some job f completed by Γ_1. If this virtual charging scheme satisfies the property that every job f completed by Γ_1 receives a total charge of at most cv_f, then we succeed in showing that Γ_1 has a competitive ratio of at most c. Designing an ingenious virtual charging scheme is crucial to the competitive analysis. In the following, we will design different virtual charging schemes to obtain the competitive ratio of Γ_1 for the restart model and the resume model respectively.

As we use a parameter β in the priority function of mechanism Γ_1, we first derive competitive ratios as functions of β. We will specify later (in Section 4.3) how to choose a suitable β (with respect to κ) to optimize the performance of Γ_1, and derive competitive ratios in terms of κ.

Here, we introduce some notation which will be used in both Section 4.1 and Section 4.2. Denote by $(1, 2, \ldots, F)$ the sequence of jobs completed by Γ_1 over time. For each job f in this sequence, let t_f be the time when job f is completed, and for convenience denote $t_0 = 0$. Divide the time into $F + 1$ intervals $I_f = [t_{f-1}, t_f)$, $f = 1, 2, \ldots, F$, and $[t_F, +\infty)$.

4.1 Analysis of the Restart Model

We study the restart model first. We assume, without loss of generality, that the optimal allocation *opt* does not interrupt any allocation, since all interrupted jobs are non-resumable. We have the following theorem.

Theorem 4.1. *For the restart model, Γ_1 has a competitive ratio of $\frac{1}{1-\beta} + \frac{1}{\beta^\kappa} + 1$.*

Proof. We introduce the virtual charging scheme as follows. For any completed job j in *opt*, if it is also completed in mechanism Γ_1, then its value is charged to itself.

Otherwise (i.e., job j is not completed by Γ_1), we consider the time s_j at which j begins execution in *opt*. Note that *opt* does not interrupt any allocation, so j is exactly allocated the time period $[s_j, s_j + l_j)$. Then s_j must be in some time interval I_f(recall $I_f = [t_{f-1}, t_f)$), and we charge the value of j to f. Define $\sigma_j := t_f - s_j$ to be the time amount between s_j and t_f. As job j is feasible at time s_j, according to Lemma 4.2, we know that the priority jobs j at time s_j is at most $v_f \beta^{t_f - s_j} = v_f \beta^{\sigma_j}$; in the meanwhile, the priority of j at time s_j is $v_j \beta^{l_j}$. We have $v_j \beta^{l_j} \le v_f \beta^{\sigma_j}$, i,e., $v_j \le v_f \beta^{\sigma_j - l_j}$. We defer the formal statement and the proof of Lemma 4.2 to the end of this subsection.

We now calculate the maximum total value charged to a completed job f in Γ_1. In the time interval I_f, denote by $(1, 2, \ldots, m)$, the sequence of jobs in *opt* whose starting time s_j belongs to I_f and ordered as $s_1 > s_2 > \cdots > s_m$. Remember that we define $\sigma_j := t_f - s_j$ to be the time amount between s_j and t_f. Then it is clear that we have $0 < \sigma_1 < \sigma_2 < \cdots < \sigma_m$ and $\sigma_j - l_j \ge \sigma_{j-1}$ for $2 \le j \le m$, since j is allocated and completed during time interval $[s_j, s_{j-1}]$. Furthermore, as the job lengths are normalized, i.e., $1 \le l_j \le \kappa$, we can deduce that:

$$\sigma_j \ge \begin{cases} 0 & \text{for } j = 1 \\ j-1 & \text{for } j \ge 2. \end{cases} \tag{5}$$

Recall that $\beta < 1$ and f may also be completed in *opt*. Therefore the total charge to job f is at most $v_f + \sum_{j=1}^{m} v_j$, which is upper bounded by

$$v_f + v_f \sum_{j=1}^{m} \beta^{\sigma_j - l_j} \le v_f(1 + \beta^{-l_1} + \sum_{j=2}^{m} \beta^{\sigma_{j-1}}) \le v_f(1 + \beta^{-l_1} + \sum_{j=1}^{m-1} \beta^{\sigma_j}) \le v_f(1 + \beta^{-\kappa} + \sum_{j=0}^{\infty} \beta^j).$$

This shows that mechanism Γ_1 is $(\frac{1}{1-\beta} + \frac{1}{\beta^\kappa} + 1)$-competitive. □

Actually, the competitive ratio obtained in this way is tight, i.e., the ratio $\frac{1}{1-\beta} + \frac{1}{\beta^\kappa} + 1$ is best possible for Γ_1. We give an example in Appendix B to show tightness.

Lemma 4.2. *For any time point $s_j \in I_f$, if job j ($\neq f$) is feasible at time s_j, then the priority of j at s_j is at most $v_f \beta^{t_f - s_j}$. Moreover, the value of j, v_j, is at most $v_f \beta^{t_f - s_j - l_j}$.*

Proof. Note that, s_j is in time interval I_f, and according to the definition of I_f, we know that f is the unique job that is completed in I_f by Γ_1. Now we prove the lemma by enumerating all possible cases.

(1) If the executing job at s_j is job f, then we know that the priority of job f at time s_j is exactly $v_f \beta^{t_f - s_j}$ (because the priority of job f at time t_f is v_f). Clearly, the priority of j at s_j is not larger than that of job f, and thus not larger than $v_f \beta^{t_f - s_j}$.

(2) If the executing job at s_j is not job f, then we assume that Γ_1 executes job $j_1, \ldots, j_k$ and f successively[13] in the time period $[s_j, t_f)$, where $k \geq 1$. Since f is the unique job completed in I_f, we can deduce that: j_1 is preempted by j_2, j_2 is preempted by j_3,...,j_k is preempted by f, and finally f is completed at time t_f. Denote $\tau_1, \ldots, \tau_k$ as the time points at which $j_1, \ldots, j_k$ are preempted respectively. We also denote $f_j(t)$ as the priority of job j at time t. We now use backward induction: First, we know that the priority of job j_k at τ_k is not larger than that of job f, i.e., $f_{j_k}(\tau_k) \leq v_f \beta^{t_f - \tau_k}$. Then, since j_{k-1} is preempted by j_k at τ_{k-1}, we know that the priority of j_{k-1} at τ_{k-1} is not larger than that of j_k. Hence, we have $f_{j_{k-1}}(\tau_{k-1}) \leq f_{j_k}(\tau_{k-1}) = f_{j_k}(\tau_k)\beta^{\tau_k - \tau_{k-1}} \leq v_f \beta^{t_f - \tau_{k-1}}$. And eventually, we can get that $f_{j_1}(\tau_1) \leq v_f \beta^{t_f - t_1}$. Since j_1 is executed at time s_j, we can deduce that $f_{j_1}(s_j) \leq v_f \beta^{t_f - s_j}$. Clearly, the priority of j at time s_j (i.e., $v_j \beta^{l_j}$) is not larger than that of j_1, thus not larger than $v_f \beta^{t_f - s_j}$.

By arranging $v_j \beta^{l_j - e_j(s_j)} \leq v_f \beta^{t_f - s_j}$, we can get $v_j \leq v_f \beta^{t_f - s_j - l_j + e_j(s_j)} \leq v_f \beta^{t_f - s_j - l_j}$, where $e_j(s_j) \geq 0$ is the valid active time of job j at time s_j. □

Some remarks on Lemma 4.2: (1) Because f is the unique job completed by Γ_1 in the time interval I_f, the priorities of the executing jobs monotonically increase during I_f. (2) Lemma 4.2 applies in both the restart model and resume model. (3) Lemma 4.2 provides a useful tool to relate the priority of a feasible job (j) at some time point ($s_j \in I_f$) to the completed job f.

4.2 Analysis of the Resume Model

Compared with the restart model, the competitive analysis for the resume model is much more complicated, because in the resume model, a job can be executed in several disjointed time intervals. The charging scheme used in the previous subsection no longer works, and we need to design a new virtual charging scheme.

Before introducing the new virtual charging scheme, we introduce some notation that will be used in this subsection. Let $\pi(j)$ denote the number of disjoint time segments allocated to a completed job j in *opt*, and $s_j^1, s_j^2, \ldots, s_j^{\pi(j)}$ denote the corresponding starting time of each segment.

We say an allocation contains a *violation* if there exist two completed jobs i and j, each of which has two segments with starting time s_i^a, s_i^c and s_j^b, s_j^d such that $s_i^a < s_j^b < s_i^c < s_j^d$. An allocation is called *standard* if it does not contain a violation. This means if an allocation is standard, for any completed job, if its starting time of execution is between two segments of another job's allocation, then its completion time is also in the same time interval (i.e., between the same two segments). We provide an obvious yet useful fact for the offline optimal allocation below.

Claim 4.3. *There exists an optimal allocation that is standard.*

For the detailed proof, please refer to Appendix C. Without loss of generality, we assume that the optimal allocation *opt* is standard.

Claim 4.4 presents an important property of the standard allocation, which will be used in the following proofs.

13. Here, j_1 can be job j, which does not affect the analysis.

Claim 4.4. *Under the execution of opt, if a job j's execution-starting time is between two segments of another job's allocation, then job j's completion time is also in the same time interval (i.e., between the same two segments).*

To analyze the competitive ratio of Γ_1 for the resume model, we propose two new virtual charging schemes (referred to as *integral charging scheme* and *segmental charging scheme*, respectively). In the *integral charging scheme*, we charge the whole value of job j in the optimal allocation *opt* to some job completed by mechanism Γ_1; while in the *segmental charging scheme*, we charge the value of j by segment, and different segments of the same job may be charged to different jobs completed by mechanism Γ_1. By using these two schemes, in Theorem 4.5 we upper bound the competitive ratio of mechanism Γ_1 by $\frac{\beta^{-\kappa}}{1-\beta}+1$ and $\frac{1}{\beta^{\kappa}}+\frac{-2}{\beta\ln\beta}+1$ respectively. As discussed in Section 4.3, the two ratios work for situations with different κ values, i.e., the first one works well for small κ and the second one works well for large κ.

Theorem 4.5. *For the resume model, the competitive ratio of* Γ_1 *is at most* $\frac{\beta^{-\kappa}}{1-\beta}+1$. *In particular, if* β *satisfies* $\kappa\beta^{\kappa}\geq\beta$, *the competitive ratio of* Γ_1 *is at most* $\min\{\frac{\beta^{-\kappa}}{1-\beta}+1,\ \frac{1}{\beta^{\kappa}}+\frac{-2}{\beta\ln\beta}+1\}$.

The proof of the theorem will be given in Section 4.2.1 and Section 4.2.2.

4.2.1 Integral Charging Scheme

Remember that we denote $(1,2,\ldots,F)$ as the sequence of jobs completed by Γ_1 over time. For each job f in this sequence, we denote the t_f as the time that job f is completed.

In the integral charging scheme, we restrict the total number of jobs (excluding f itself) that charged to job f: we does not allow this number to exceed $\lfloor t_f-t_{f-1}\rfloor+1$. In particular, we introduce the notation of "*saturation* " in Definition 4.6.

Definition 4.6 (**Saturated**)**.** *For any job* f*, if the number of jobs (excluding* f *itself) charged to* f *is less than* $\lfloor t_f-t_{f-1}\rfloor+1$*, we say that* f *is* unsaturated*; otherwise* f *is* saturated.

Let W denote the set of jobs completed by *opt*, and $W_f\subseteq W$ denote the set of jobs $j\in W$ with $s_j^1\in I_f$. Let A denote the set of jobs in W whose values have already been charged to some jobs completed by Γ_1.

The integral charging scheme is described in Scheme 1. For simplicity, we refer to Line $1-2$ as Step 1, Line $4-11$ as Step 2, and Line $12-21$ as Step 3.

Here we give some intuitive explanations about Step 2 and Step 3.

In Step 2, for each job f $(f=1,\ldots,F)$, we pick at most $\lfloor t_f-t_{f-1}\rfloor+1$ jobs from W_f and charge their values to f. The rule of picking jobs follows "largest s_j^1 first", and the k-th picked job[14] with s_j^1 no later than t_f-k+1.

14. By slight abuse of notations, we still denote it as job j, and thus the start time of its first segment is s_j^1.

SCHEME 1: Integral Charging Scheme

1: **Initial:** $A \leftarrow \emptyset$.
2: For any job in W, if it is also completed by mechanism Γ_1, charge its value to itself, and add it to A.
3: **while** $W \setminus A \neq \emptyset$, **do**
4:   **for** $f = 1$ to F, **do**
5:     **for** $k = 0$ to $\lfloor t_f - t_{f-1} \rfloor$, **do**
6:       $J^k := \{j' | \ (s^1_{j'} \leq t_f - k) \wedge (j' \in W_f \setminus A)\}$;
7:       **if** $J^k \neq \emptyset$, **then**
8:         Set $j = \arg\max_{j' \in J^k}(s^1_{j'})$, add j to A, and charge its value to f.
9:       **end if**
10:     **end for**
11:   **end for**
12:   **for** $f = F$ to 1, **do**
13:     **while** $W_f \setminus A \neq \emptyset$, **do**
14:       Set $j = \arg\max_{j' \in W_f \setminus A}(s^1_{j'})$, and add j to A;
15:       **if** $s_j^{\pi(j)} \in I_{f+h_j}$ for some $0 \leq h_j \leq F - f$, **then**
16:         Charge j's value to the unsaturated job with smallest completion time in the set $\{f+1, \ldots, f+h_j\}$;
17:       **else if** $s_j^{\pi(j)} \in [t_F, +\infty)$, **then**
18:         Charge j's value to the unsaturated job with smallest completion time in the set $\{f+1, \ldots, F\}$;
19:       **end if**
20:     **end while**
21:   **end for**
22: **end while**

In Step 3, we consider jobs (in W) whose values are not charged to any job in the first two steps. Consider a job j with s^1_j located in interval I_f and $s_j^{\pi(j)}$ located in I_{f+h_j} (or $[t_F, +\infty)$). We charge its value to an unsaturated job in the job set $\{f+1, \ldots, f+h_j\}$ (or $\{f+1, \ldots, F\}$). The rule of selecting the unsaturated job follows "smallest completion time first".

We will show that after three steps all jobs in W are charged to some completed jobs in Γ_1 (see Claim 4.9). First, we give two observations below.

Observation 4.7. *In the integral charging scheme, for any job $f \in \{1, 2, \ldots, F\}$ and any time $t \in I_f$, the number of jobs charged to f with their start time in opt being in $[t, t_f)$ (charged at step 2) is no more than $\lfloor t_f - t \rfloor + 1$.*

Observation 4.8. *In the integral charging scheme, for any job $f \in \{1, 2, \ldots, F\}$ completed by mechanism Γ_1, the total number of jobs charged to f (excluding f itself) is at most $\lfloor t_f - t_{f-1} \rfloor + 1$.*

Observation 4.7 is derived from Lines 5-6 in Scheme 1, and Observation 4.8 is derived from the restriction that a saturated job can not be charged any more.

Claim 4.9. *In the integral charging scheme, all jobs in W have been charged to some jobs completed by mechanism Γ_1.*

Proof. Suppose on the contrary that there exists $i \in W_f$ such that i is not charged to any job in $\{f, f+1, \dots, f+h_i\}$. Here, we introduce a notation $e_i^*(t) = \int_0^t \mu(opt(s) = i)ds$ to denote the valid active time of resumable job i at time t in *opt*. Since the length of every job is at least 1,[15] there exists an allocation segment $[s', s'']$ of job i such that $e_i^*(s') < 1, e_i^*(s'') \geq 1$, and $opt(t) = i$ for any $t \in [s', s'']$. Suppose s'' belongs to I_{f+h}. By the definition of h_i, we have $h \leq h_i$.

According to the assumption, we know: (a) i is not charged to f. (b) All jobs in $\{f+1, f+2, \dots, f+h\}$ have been saturated in the above charging process when we charge job i.

From point (a), we can deduce that in Step 2, there are at least $\lfloor t_f - s_i^1 \rfloor + 1$ jobs (whose values are charged to f) with $s_j^1 \in (s_i^1, t_f]$ (by Observation 4.7). Otherwise i would be charged to f in Step 2. We denote J_a as the set of these $\lfloor t_f - s_i^1 \rfloor + 1$ jobs.

As for point (b), recall that a job f' ($f' \in \{f+1, \dots, f+h\}$) is *saturated* if there are $\lfloor t_i - t_{i-1} \rfloor + 1$ jobs whose values are charged to f' (see Definition 4.6). Hence, we can deduce that there are at least $(\lfloor t_{f+1} - t_f \rfloor + 1) + \cdots + (\lfloor t_{f+h} - t_{f+h-1} \rfloor + 1)$ jobs (whose values are charged to $\{f+1, \dots, f+h\}$) with their starting time satisfying $s_j^1 \in (s_i^1, t_{f+h}]$. In particular, among these jobs, there are at most $(\lfloor t_{f+h} - s'' \rfloor + 1)$ jobs with $s_j^1 \in (s'', t_{f+h}]$ (whose value must be charged to $f+h$).[16] Therefore, we can deduce that there are at least

$$(\lfloor t_{f+1} - t_f \rfloor + 1) + \cdots + (\lfloor t_{f+h} - t_{f+h-1} \rfloor + 1) - (\lfloor t_{f+h} - s'' \rfloor + 1)$$

jobs (whose values are charged to $\{f+1, \dots, f+h\}$) with $s_j^1 \in (s_i^1, s'']$ (denote J_b as the set of these jobs).

Note that $J_a \cap J_b = \emptyset$, as all jobs in J_a are charged to f, while all jobs in J_b are charged to $\{f+1, \dots, f+h\}$. Therefore, we deduce that the number of jobs with start time contained in $(s_i^1, s'']$ is at least $|J_a| + |J_b|$, i.e.,

$$\begin{aligned} &(\lfloor t_f - s_i^1 \rfloor + 1) + (\lfloor t_{f+1} - t_f \rfloor + 1) + \cdots + (\lfloor t_{f+h} - t_{f+h-1} \rfloor + 1) - (\lfloor t_{f+h} - s'' \rfloor + 1) \\ >&(t_{f+h} - s_i^1) - (\lfloor t_{f+h} - s'' \rfloor + 1) \geq \lfloor s'' - s_i^1 \rfloor - 1. \end{aligned} \tag{6}$$

So, there are more than $\lfloor s'' - s_i^1 \rfloor - 1$ jobs different from i in $[s_i^1, s'']$. Recall that we assume *opt* is standard, hence, these jobs are entirely scheduled in (s_i^1, s''), i.e., all time segments of such a job are allocated in (s_i^1, s'') (Claim 4.4). Since the length of every job is at least 1, we reach a contradiction.

□

According to the integral charging scheme, the charges to a completed job f have three origins, corresponding to the three steps in Scheme 1. From Step 1, obviously, the charge to job f is at most v_f. We now calculate the maximum total charge from Step 2.

15. As stated in the Problem Formulation section, we assume job lengths are located in $[1, \kappa]$ for simplicity. However, by scaling, all our results and proofs can be easily generalized to the case of $[l_{\min}, \kappa \cdot l_{\min}]$, where l_{min} is the shortest length of jobs.
16. Because: (i) in Step 2, there might be at most $(\lfloor t_{f+h} - s'' \rfloor + 1)$ jobs with $s_j^1 \in (s'', t_{f+h}]$ which could be charged to $f+h$; (ii) in Step 3, the jobs with $s_j^1 \in I_{f+h}$ could not be charged to $f+h$.

Suppose the total number of jobs charged to f from Step 2 is m. We rename them as $1, 2, \ldots, m$ according to $\sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_m$, and claim $v_j \leq v_f \beta^{\sigma_j - l_j}$ (Lemma 4.2 is used here), where $\sigma_j := t_f - s_j^1$, for $1 \leq j \leq m$. According to the rule of picking jobs in Step 2, we have $\sigma_j \geq j - 1$. So it is clear that the sum of values of all these m jobs is at most

$$v_f \sum_{j=1}^{m} \beta^{\sigma_j - l_j} \leq v_f \sum_{j=1}^{m} \beta^{j-1-\kappa} = v_f \sum_{j=0}^{m} \beta^{j-\kappa}. \tag{7}$$

It remains to calculate the maximum total charge from Step 3. According to Observation 4.8, we know that the number of jobs charged to f from Step 3 is at most $\lfloor t_f - t_{f-1} \rfloor + 1 - m$. Now we need to bound the value of each such job j. The key is to build a relationship between its value and the value of job f. However, according to the charging rule in Step 3, the start time s_j^1 of job j is not located in the time interval I_f. In this case, we cannot use Lemma 4.2 directly to derive an inequality like $v_j \leq v_f \beta^{\sigma_j - l_j}$. Because it remains to check whether j is feasible at t_{f-1} (note that t_{f-1} is the left endpoint of time interval I_f).

We define the *critical time* of a job as $t_j^* := d_j - l_j$. If we can prove that $t_j^* \geq t_{f-1}$, then job j must be feasible at time t_{f-1} for Γ_1. Thus, by applying Lemma 4.2, we can easily get

$$v_j \leq v_f \beta^{t_f - t_{f-1} - l_j} \leq v_f \beta^{t_f - t_{f-1} - \kappa}. \tag{8}$$

Fortunately, the following lemma shows that $t_j^* \geq t_{f-1}$ holds.

Lemma 4.10. *According to the charging scheme, if $j \in W_f$ is charged to a completed job $f + k$ (where $1 \leq k \leq h_j$), then the critical time of job j satisfies $t_j^* \geq t_{f+k-1}$.*

Proof. We prove the lemma by contradiction and suppose $t_j^* < t_{f+k-1}$. Then the total length of all the other jobs whose *opt* allocation is between s_j^1 and $s_j^{\pi(j)}$ is $(s_j^{\pi(j)} + l_j^{\pi(j)}) - s_j^1 - l_j$, which is at most

$$d_j - s_j^1 - l_j = (d_j - l_j) - s_j^1 = t_j^* - s_j^1 < t_{f+k-1} - s_j^1. \tag{9}$$

Since j is charged to $f+k$, from Step 3 we know that all jobs in $\{f+1, f+2, \ldots, f+k-1\}$ are saturated. Thus, there are at least

$$(\lfloor t_f - s_j^1 \rfloor + 1) + (\lfloor t_{f+1} - t_f \rfloor + 1) + \cdots + (\lfloor t_{f+k-1} - t_{f+k-2} \rfloor + 1) \geq \lfloor t_{f+k-1} - s_j^1 \rfloor + 1 \tag{10}$$

jobs whose start time belongs to the interval (s_j^1, t_{f+k-1}).

Recall that *opt* is standard. Hence, all these jobs' allocated time segments are between the first segment and the last segment of job j (according to Claim 4.4), Equation (9) and Equation (10) constitute a contradiction since every job's length is at least 1. $\square$

Combining the analysis above, we know that: (1) the total charge to f from Step 1 is at most v_f; (2) assuming m jobs are charged to f from Step 2, the total charge from these m jobs is at most $v_f \sum_{j=0}^{m} \beta^{j-\kappa}$ according to Equation (7); (3) the number of jobs charged to f from Step 3 is at most $\lfloor t_f - t_{f-1} \rfloor + 1 - m$ according to Definition 4.6, and the value of each job is at most $v_f \beta^{t_f - t_{f-1} - \kappa}$ according to Equation (8). Therefore, the total charge to f is at most

$$v_f + v_f \sum_{j=0}^{m-1} \beta^{j-\kappa} + (\lfloor t_f - t_{f-1} \rfloor + 1 - m) v_f \beta^{t_f - t_{f-1} - \kappa} \leq v_f (1 + \beta^{-\kappa} \sum_{j=0}^{\lfloor t_f - t_{f-1} \rfloor + 1} \beta^j),$$

which is upper bounded by $v_f(1 + \frac{\beta^{-\kappa}}{1-\beta})$, indicating that the competitive ratio of mechanism Γ_1 is upper bounded by $\frac{\beta^{-\kappa}}{1-\beta} + 1$.

4.2.2 **Segmental Charging Scheme**

Recall that we use $s_j^1, s_j^2, \ldots, s_j^{\pi(j)}$ to denote the starting time of all time segments allocated to job j in *opt*. Let $\Delta_j^1, \Delta_j^2, \ldots, \Delta_j^{\pi(j)}$ denote those time segments, and $l_j^1, l_j^2, \ldots, l_j^{\pi(j)}$ denote the length of them.

In the segmental charging scheme, each segment Δ_j^k is given a value $\rho_j l_j^k$, in which $\rho_j := \frac{v_j}{l_j}$ is the value density of job j. We describe the segmental charging scheme in Scheme 2. For simplicity, we refer to Line $2-3$ as Type-1 charge, Line $4-5$ as Type-2 charge, and Line $6-7$ as Type-3 charge.

SCHEME 2: Segmental Charging Scheme

1: **for** each segment Δ_j^k in *opt* **do**
2:   **if** mechanism Γ_1 also completes j by its deadline, **then**
3:     Charge the value $\rho_j l_j^k$ to j.
4:   **else if** $s_j^k \in I_f$ for some $f \in \{1, 2, , \ldots, F\}$, and $\rho_j \leq v_f \beta^{\sigma_j - 1}$, where $\sigma_j := t_f - s_j^k$, **then**
5:     Charge the value $\rho_j l_j^k$ to f.
6:   **else**
7:     Charge $\rho_j l_j^k$ to f^*, where f^* is the first job completed by Γ_1 from time t_j^* on, where t_j^* is the critical time of job j.
8:   **end if**
9: **end for**

It is clear that the Type-1 charge received by a job f is at most v_f. Next, we bound the Type-2 and Type-3 charges.

Lemma 4.11. *The total Type-2 charge that a job f receives is at most $-\frac{v_f}{\beta \ln \beta}$.*

Proof. Let R_2 denote the set of job segments whose charges to f are Type-2. For each $\Delta_j^k \in R_2$, the charge from it is $\rho_j l_j^k$. And from Line 4 in Scheme 2, we know $\rho_j \leq v_f \beta^{\sigma_j - 1}$, where $\sigma_j = t_f - s_j^k$. Thus the total Type-2 charge is at most

$$\sum_{\Delta_j^k \in R_2} \rho_j l_j^k \leq v_f \sum_{\Delta_j^k \in R_2} \beta^{\sigma_j - 1} l_j^k \leq v_f \sum_{\Delta_j^k \in R_2} \int_{\sigma_j - l_j^k}^{\sigma_j} \beta^{x-1} dx \leq v_f \int_0^\infty \beta^{x-1} dx \leq -\frac{v_f}{\beta \ln \beta},$$

where the second inequity holds by $\beta < 1$. Therefore, f receives a total Type-2 charge of at most $-\frac{v_f}{\beta \ln \beta}$. □

In the following, we study the Type-3 charge and denote R_3 as the set of job segments which constitute Type-3 charges to f.

First, we claim that, if β satisfies some condition, then we can get $[s_j^k, s_j^k + l_j^k] \subseteq [t_f, t_f + l_j] \subseteq [t_f, t_f + \kappa]$ for each $\Delta_j^k \in R_3$ (Claim 4.12).

Claim 4.12. *If β satisfies the function: $g(x) = x\beta^x \geq \beta$ for $1 \leq x \leq \kappa$; then we have $[s_j^k, s_j^k + l_j^k] \subseteq [t_f, t_f + l_j]$, for each $\Delta_j^k \in R_3$.*

Proof. To prove $[s_j^k, s_j^k + l_j^k] \subseteq [t_f, t_f + l_j]$, we only need to prove the inequality below:

$$t_f \leq s_j^k \leq t_f + l_j - l_j^k. \tag{11}$$

The inequality $s_j^k \leq t_f + l_j - l_j^k$ holds because $(s_j^k + l_j^k) - l_j \leq d_j - l_j = t_j^* \leq t_f$.

Next we prove $t_f \leq s_j^k$. Suppose $s_j^k \in I_{f'}$ for some f' ($I_{f'}$ is not later than I_f, and might equal I_f). Then according to the Type-3 charging rule, we have $\rho_j = \frac{v_j}{l_j} > v_{f'}\beta^{\sigma'_j - 1}$, where $\sigma'_j = t_f - s_j^k$.

We now use the condition for β: $g(x) = x\beta^x \geq \beta$ for $1 \leq x \leq \kappa$. Then we have $l_j\beta^{l_j} \geq \beta$, hence $\frac{v_j}{l_j} \leq v_j\beta^{l_j - 1}$. Combining the above two inequalities ($\frac{v_j}{l_j} > v_{f'}\beta^{\sigma'_j - 1}$ and $\frac{v_j}{l_j} \leq v_j\beta^{l_j - 1}$), we have $v_j\beta^{l_j} > v_{f'}\beta^{\sigma'_j}$, thus $v_j > v_{f'}\beta^{\sigma'_j - l_j}$, which contradicts the fact that f' is completed at $t_{f'}$ with priority $v_{f'}$ (Lemma 4.2 is used here). Therefore, we have $t_f \leq s_j^k$. □

By Claim 4.12, we know the allocation of all the segments with Type-3 charges to f are in a restricted interval $[t_f, t_f + \kappa]$. Hence, we can derive that $\sum_{\Delta_j^k \in R_3} l_j^k \leq \kappa$.

Lemma 4.13. *If β satisfies the function: $g(x) = x\beta^x \geq \beta$ for $1 \leq x \leq \kappa$; then the total Type-3 charge that a job f receives is at most $v_f(\frac{-1}{\beta \ln \beta} + \beta^{-\kappa})$.*

Proof. According to the Type-3 charging rule, j is not completed by the mechanism; if we consider the critical point of j, i.e., t_j^* (in time interval I_f), then by applying Lemma 4.2, we can deduce that $v_j\beta^{l_j} \leq v_f\beta^{t_f - t_j^*} \leq v_f$. Therefore we have $\frac{v_j}{l_j} \leq \frac{v_f}{l_j\beta^{l_j}}$. Now we can bound the total Type-3 charge that f receives

$$\sum_{\Delta_j^k \in R_3} \rho_j l_j^k = \sum_{\Delta_j^k \in R_3} \frac{v_j}{l_j} l_j^k \leq \sum_{\Delta_j^k \in R_3} \frac{v_f}{l_j\beta^{l_j}} l_j^k = v_f \sum_{\Delta_j^k \in R_3} \frac{l_j^k}{g(l_j)}, \tag{12}$$

Note that the function $g(l_j) = l_j\beta^{l_j}$ is increasing for $1 \leq l_j \leq \frac{-1}{\ln \beta}$ and decreasing for $l_j \geq \frac{-1}{\ln \beta}$. So we have

$$g(l_j) \geq \begin{cases} \beta & \text{for } 1 \leq l_j \leq \frac{-1}{\ln \beta} \\ \kappa\beta^\kappa & \text{for } \frac{-1}{\ln \beta} \leq l_j \leq \kappa, \text{ if } \kappa > \frac{-1}{\ln \beta}. \end{cases} \tag{13}$$

By Claim 4.12, we know $[s_j^k, s_j^k + l_j^k] \subseteq [t_f, t_f + l_j] \subseteq [t_f, t_f + \kappa]$ for each $\Delta_j^k \in R_3$. Therefore, on the one hand, for each $\Delta_j^k \in R_3$ with $\frac{-1}{\ln \beta} \leq l_j \leq \kappa$ (denote this set as R_3^a), we have $\sum_{\Delta_j^k \in R_3^a} l_j^k \leq \kappa$; on the other hand, for each $\Delta_j^k \in R_3$ with $l_j \leq \frac{-1}{\ln \beta}$ (denote this set as R_3^b), we have $\sum_{\Delta_j^k \in R_3^b} l_j^k \leq \frac{-1}{\ln \beta}$.

Then, (12) becomes

$$\begin{aligned} \sum_{\Delta_j^k \in R_3} \rho_j l_j^k \leq v_f \sum_{\Delta_j^k \in R_3} \frac{l_j^k}{g(l_j)} &= v_f \Big(\sum_{\Delta_j^k \in R_3^a} \frac{l_j^k}{g(l_j)} + \sum_{\Delta_j^k \in R_3^b} \frac{l_j^k}{g(l_j)} \Big) \\ &\leq v_f \Big(\frac{\sum_{\Delta_j^k \in R_3^a} l_j^k}{\kappa \beta^\kappa} + \frac{\sum_{\Delta_j^k \in R_3^b} l_j^k}{\beta} \Big) \leq v_f \Big(\frac{\kappa}{\kappa \beta^\kappa} + \frac{\frac{-1}{\ln \beta}}{\beta} \Big), \end{aligned} \tag{14}$$

which means the Type-3 charge is bounded by $v_f(\frac{-1}{\beta \ln \beta} + \frac{1}{\beta^\kappa})$. □

Based on Lemmas 4.11 and 4.13, we can obtain that when $\kappa \beta^\kappa \geq \beta$,[17] the total charge to a job f completed by mechanism Γ_1 is at most $v_f(\frac{1}{\beta^\kappa} + \frac{-2}{\beta \ln \beta} + 1)$. This implies that the competitive ratio of mechanism Γ_1 is upper bounded by $\frac{1}{\beta^\kappa} + \frac{-2}{\beta \ln \beta} + 1$.

4.3 Discussions

An advantage of our mechanism is that it can handle the settings with different values of κ in a unified framework. We only need to set parameter β to different values in Theorem 4.1 and Theorem 4.5 so as to adapt to different settings of job lengths (as shown in the following corollaries).

Corollary 4.14. *By setting $\beta = 1 - (1-\epsilon)^2 \cdot \frac{\ln \kappa}{\kappa}$, where $\epsilon > 0$ is an arbitrary small constant, mechanism Γ_1 achieves a competitive ratio $(\frac{1}{(1-\epsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln \kappa}$ for the restart model and a competitive ratio $(\frac{2}{(1-\epsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln \kappa}$ for the resume model.*

The proof can be found in Appendix D. As for Corollary 4.14, we have the following discussions:

(1) For the restart model, mechanism Γ_1 achieves a competitive ratio of $(\frac{1}{(1-\epsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln \kappa}$, which improves upon the best-known algorithmic result $\frac{6\kappa}{\log \kappa} + O(\kappa^{\frac{5}{6}})$ (Ting, 2008) for the standard online scheduling without strategic behavior.

(2) For the resume model, when κ is large, mechanism Γ_1 achieves a competitive ratio of $(\frac{2}{(1-\epsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln \kappa}$, which is slightly worse than the result obtained for the restart model (within a factor of 2). Asymptotically speaking, Γ_1 is near optimal, since its competitive ratio has the same order (w.r.t. κ) as the lower bound shown in Theorem 2.5. Furthermore, our analysis generalizes the results obtained by Durr et al. (2012) to the continuous value of time and the strategic setting.

(3) When κ is relatively small, the ratios given in Corollary 4.14 will become loose. In particular, when κ approaches 1, the above ratios will approach infinity since $\ln \kappa$ approaches 0. In this case, we need a different setting of β (see Corollary 4.15).

17. Note that, the function $g(x) = x\beta^x$ is increasing for $1 \leq x \leq \frac{-1}{\ln \beta}$ and decreasing for $x \geq \frac{-1}{\ln \beta}$. Therefore, we only need to require $\kappa \beta^\kappa \geq \beta$, and we can naturally derive $g(x) = x\beta^x \geq \beta$ for $1 \leq x \leq \kappa$.

Corollary 4.15. *By choosing $\beta = \frac{\kappa}{\kappa+1}$, the competitive ratio of mechanism Γ_1 is $\kappa + 2 + (1+\frac{1}{\kappa})^\kappa < \kappa + 2 + e$ for the restart model and $(\kappa+1)(1+\frac{1}{\kappa})^\kappa + 1$ for the resume model.*

Similarly, we have the following discussions:

(1) The competitive ratio of Γ_1 is linear in κ, since $(1+\frac{1}{\kappa})^\kappa$ is bounded by e.

(2) In particular, when $\kappa = 1$, the ratios in the above corollary become 5 for both the restart and resume model, which matches the lower bound given in Theorem 2.4. In this regard, we say that Γ_1 is optimal. On the other hand, this also shows that the lower bound of 5 in Theorem 2.4 is tight.

5. Conclusion and Future Work

In this paper, we studied the online scheduling problem in a strategic setting. As summarized in Table 2, we proved that in both the restart model and the resume model, the competitive ratio of any IC mechanism cannot be less than 5 when $\kappa = 1$ and cannot be less than $\frac{\kappa}{\ln \kappa} + 1 - o(1)$ for large κ. We designed a simple IC mechanism Γ_1 to schedule jobs on a single machine and proved that it has near optimal approximation guarantees (in terms of social efficiency) in both the restart model and the resume model through competitive analysis: as shown in Table 2, the mechanism is optimal in terms of competitive ratio in both the restart model and the resume model when $\kappa = 1$, and it is near optimal for the restart model when κ is large enough.

Table 2: Summary of bounds on competitive ratio

Model	Restart Model		Resume Model	
	$\kappa = 1$	asymptotic κ	$\kappa = 1$	asymptotic κ
LB for any IC mech.	5	$\frac{\kappa}{\ln \kappa} + 1 - o(1)$	5	$\frac{\kappa}{\ln \kappa} + 1 - o(1)$
UB of the proposed mech.	5	$(\frac{1}{(1-\epsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln \kappa}$	5	$(\frac{2}{(1-\epsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln \kappa}$

In proving the lower bounds, we introduce the shadow job argument which reflects the IC constraint. This argument is very helpful in extending bounds in non-strategic setting to strategic setting. The second contribution of this work is that we design several virtual charging schemes to analyze the competitive ratio of our mechanism. The ideas of these virtual charging schemes are of methodological significance and may be used to address other problems.

There are multiple directions to explore in the future.

It is an interesting problem whether an IC competitive mechanism can be designed for the hybrid model, in which there exist both resumable and non-resumable jobs. Many new strategic issues may arise in the hybrid model. For example, can a resumable job disguise as a non-resumable job to get better off?

Another open problem is whether a tighter competitive analysis of Γ_1 can be made for the resume model. Our conjecture is that the competitive ratio obtained by Γ_1 has an uniform form: $\frac{1}{1-\beta} + \frac{1}{\beta^\kappa} + 1$, for both the restart model and the resume model.

Furthermore, given the popularity of cloud computing in today's IT industry, it is of practical importance to extend our work to the setting of job scheduling on multiple heterogeneous machines.

Appendix A. Algorithms for the Critical-Value Payment

Please note that the *critical time point* in Algorithm 3 and Algorithm 4 means the time point when some new jobs arrive or some existing jobs are completed.

Algorithm 3: Compute the critical-value payment in the restart model

---

for each job j which is completed **do**
  Run **Algorithm** 1 without job j. Let T be the set of all critical time points $t \in [r_j, d_j)$.
  for every $t \in T$ **do**
    if there exists job k such that $x(t) = k$, **then** define $f_t = v_k \beta^{l_k - e_k(t)}$;
    else $f_t = 0$;
  end-for
  for every time point $t' \in T \cap [r_j, d_j - l_j)$ **do**
    Define $f^*_{t'} = \max\{f_t / \beta^{t'-t} : t \in (T \cap [t', t' + l_j))\}$;
  end-for
  Let $f^* = \min_{t'} f^*_{t'}$.
  $p_j = f^* / \beta^{l_j}$.
end

Algorithm 4: Compute the critical-value payment in the resume model

---

for each job j which is completed **do**
  Run **Algorithm** 1 without job j.
  Let $\{t_0, t_1, \ldots, t_m\}$ be the set (denoted as T) of all critical time points in $[r_j, d_j)$, and $t_0 = r_j$.
  Denote the period between two critical time point as $z_i = t_i - t_{i-1}$, where $i = 1, 2, \ldots, m$.
  for every $t_i \in T$ **do**
    if there exists job k such that $x(t_i) = k$, **then** define $f_{t_i} = v_k \beta^{l_k - e_k(t_i)}$;
    else $f_{t_i} = 0$;
  end-for
  Initially, $T^* \leftarrow \emptyset$, $h^* = 0$.
  while $h^* < l_j$ **do**
    $t' = \arg\min_{t_i \in T \setminus T^*} f_{t_i}$, ties are broken in favor of smaller t_i;
    Initially, $e' = 0$;
    for every time point $t_i \geq t'$ that satisfies $f_{t_i} \leq f_{t'} \beta^{-e'}$ **do**
      $e' = e' + z_i$; and
        if $t_i \notin T^*$,
        then add t_i to T^*, and $h^* = h^* + z_i$;
        end-for
  end-while
  Let t'_1 be the earliest critical time point in T^*. Let $t^* = \arg\max_{t_i \in T^*} f_{t_i}$.
  Denote the critical time points in T^* and between t'_1 and t^* as $t'_2, t'_3, \ldots, t'_k$.
  Denote the relevant periods of those critical time points as $z'_1, z'_2, \ldots, z'_k$, and z^*.
  $p_j = f_{t^*} / \beta^{l_j - (z'_1 + \cdots + z'_k)}$.
end

Appendix B. An Example for Analysis Tightness

Example B.1. *There are two types of jobs: long and short. The length of long jobs is* κ*, while the length of short jobs is* 1*. Let* p *be a large integer, and the number of long and short jobs are* $p+1$ *and* $p\kappa-1$ *respectively. The first long job* J_0^l *is released at time* 0*, and its type is* $\theta_0^l = (0, \kappa, \kappa, 1)$*. For* $p-2 \geq i \geq 1$*, job* J_i^l *has type* $\theta_i^l = (i(\kappa-\epsilon), (i+1)\kappa - i\epsilon, \kappa, \beta^{-i\kappa})$*. Long job* J_{p-1}^l *has type* $\theta_{p-1}^l = ((p-1)(\kappa-\epsilon), (p+2)\kappa, \kappa, \beta^{-(p-1)\kappa}+\delta)$*. Job* J_p^l *has type* $\theta_p^l = (p(\kappa-\epsilon), (p+1)\kappa - p\epsilon, \kappa, \beta^{-p\kappa+\epsilon}-\delta)$*. Here,* ϵ *and* δ *are small constants satisfying* $p\epsilon \ll 1$ *and* $\delta \ll \epsilon$*. In the meanwhile, we have short jobs as follows. For* $j = 1, \ldots, p\kappa-1$*, we denote* J_j^s *as the* j*th short job, whose type is* $\theta_j^s = (j-(p-1)\epsilon, j+1-(p-1)\epsilon, 1, \beta^{\kappa-(j+1)+(p-1)\epsilon} - \frac{\delta}{\kappa})$*. for* $j = 1, \ldots, p\kappa-1$*.*

It can be verified that only one job J_{p-1}^l can be completed in mechanism Γ_1, with a social welfare $\sim \beta^{-(p-1)\kappa}$. While in the optimal solution, all the short jobs will be completed, and after that, J_p^l and J_{p-1}^l will be completed successively, with a social welfare $\sim (\beta^{\kappa-2} + \beta^{\kappa-3} + \ldots + \beta^{-(p-1)\kappa}) + \beta^{-(p-1)\kappa} + \beta^{-p\kappa}$. Therefore, the competitive ratio of mechanism Γ_1 is at least $(\beta^{p\kappa-2} + \ldots + \beta + 1) + 1 + \beta^{-\kappa} = \frac{1-\beta^{p\kappa-1}}{1-\beta} + 1 + \beta^{-\kappa}$, which tends to $\frac{1}{1-\beta} + \frac{1}{\beta^\kappa} + 1$, when $p \to \infty$.

Appendix C. Proof of Claim 4.3

Proof. Suppose an optimal allocation *opt* is not standard, i.e., there exist a completed job i with two segments beginning at time s_i^a and s_i^c and a completed job j with two segments beginning at time s_j^b and s_j^d such that $s_i^a < s_j^b < s_i^c < s_j^d$. We now do the following process to obtain a standard optimal allocation: if the length of job j's b-th segment (denote as l_j^b) is larger than that of i's c-th segment (denote as l_i^c), we exchange i's c-th segment with j's b-th segment located in $[s_j^b, s_j^b + l_i^c]$; otherwise, we exchange j's b-th segment with i's c-th segment located in $[s_i^c + l_i^c - l_j^b, s_j^c + l_i^c]$. For all the other segments, their order remains unchanged. It is easy to see that the new allocation is still feasible and obtains the same social welfare. We do such kind of exchanges until there is no violation, and then obtain a standard optimal allocation. □

Appendix D. Proof of Corollary 4.14

Proof. For every constant $c < 1$ and large enough x, we have $(1-\frac{c}{x})^{-x} \leq e$. When κ is large enough, by choosing $\beta = 1-(1-\epsilon)^2 \cdot \frac{\ln \kappa}{\kappa}$, we have $\beta \to 1$, and

$$\beta^{-\kappa} = (1 - \frac{(1-\epsilon)^2 \ln \kappa}{\kappa})^{-\frac{\kappa}{(1-\epsilon)\ln\kappa}\cdot(1-\epsilon)\ln\kappa} \leq e^{(1-\epsilon)\ln\kappa} \leq \kappa^{(1-\epsilon)} = o(\frac{\kappa}{\ln \kappa}).$$

By using Taylor's theorem, we know

$$-\ln\beta = (1+o(1))(1-\beta) = (1+o(1)) \cdot \frac{c^2 \ln \kappa}{\kappa}.$$

Thus the competitive ratio is $\frac{1}{1-\beta} + \frac{1}{\beta^\kappa} + 1 = (\frac{1}{(1-\epsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln\kappa}$ for the restart model, and $\frac{-2}{\beta\ln\beta} + \frac{1}{\beta^\kappa} + 1 = (\frac{\kappa}{c^2 \ln \kappa}(2+o(1)) + o(\frac{\kappa}{\ln\kappa})) + 1 = (\frac{2}{(1-\epsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln\kappa}$ for the resume model, respectively. □

Appendix E. The Multiple Machines Extension

Suppose there are C identical machines, and each of them can process at most one job at any given time. Similar to the work of Lucier et al. (2013), we assume that at most h machines can be allocated to a single job at any given time. This parameter stands for a common parallelism bound of the system.

The notion of preemption is specified as follow: A job may be processed on any number of machines between 1 and h, and the number of machines allocated to this job may fluctuate, and only if the number decreases to 0, we treat this job as preempted. Thus, the notation of preemption-restart and preemption-resume can be defined accordingly.

Each job $j \in J$ is characterized by a private type $\theta_j = (r_j, d_j, s_j, v_j)$. Instead of l_j, where we use s_j to denote job's size (e.g., the number of machine hours required to complete the job). Without causing any confusion, we let κ be the maximum ratio between the sizes of any two jobs: $\kappa = \max_{i,j \in J} \frac{s_i}{s_j}$. For simplicity, we assume all job sizes fall in $[1, \kappa]$. If $\kappa = 1$, all the jobs have the identical size; otherwise they have different sizes.

E.1 A Simple Case: $h = 1$

In this case, we design a new mechanism Γ_2 based on the single-machine mechanism Γ_1. The payment rule of Γ_2 is exactly the same as Γ_1, and its allocation rule is shown in Algorithm 2, which is also similar to that of Γ_1. Since each job can be processed on at most one machine, the mechanism will choose the C jobs (if any) in $J_F(t)$ with the highest priorities $\hat{v}_i \cdot \beta^{\hat{s}_i - e_i(\hat{\theta}, t)}$ to execute. Note that here the *valid active time* of job j until time t is computed as

$$e_j(t) = \sum_{i=1}^{C} \int_{t'}^{t} \mu(x_i(s) = j) ds. \tag{15}$$

where $\mu(\cdot)$ is an indicator function, and $t' = \arg\max_{s \leq t}[\sum_{i=1}^{C} \mu(x_i(s) = j)] = 0$. That is to say, we treating resumable jobs as non-resumable jobs for simple. We summarize the theoretical properties of Γ_2 in Theorem E.1.

Algorithm 5: The allocation rule of Mechanism Γ_2

```
for all t do
    if |J_F(t)| ≥ C then
        process the C jobs with highest priorities in J_F(t);
    else process all the jobs in J_F(t).
end
```

Theorem E.1. *Mechanism Γ_2 is IC and has the following properties:*

- *In the restart model, by setting $\beta = \frac{\kappa}{(\kappa+1)}$, we can get a competitive ratio of $\kappa + 2 + (1 + \frac{1}{\kappa})^\kappa$ for Γ_2; by setting $\beta = 1 - (1-\varepsilon)^2 \cdot \frac{\ln \kappa}{\kappa}$ for arbitrary small $\varepsilon > 0$, we can get another competitive ratio of $(\frac{1}{(1-\varepsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln \kappa}$.*

- *In the resume model, by setting $\beta = 1 - (1-\varepsilon)^2 \cdot \frac{\ln \kappa}{\kappa}$ for arbitrary small $\varepsilon > 0$, we can get a competitive ration of $(\frac{2}{(1-\varepsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln \kappa}$.*

As for the above theorem, we have the following discussions.

(1) Similar to what we have done in the single-machine setting, for the restart model, we give two competitive ratios for Γ_2. When κ is small, the first ratio is better (in particular, when $\kappa = 1$, this competitive ratio becomes 5 and is thus optimal according to Theorem 2.4). When κ is large, the second ratio is better instead and is near optimal according to Theorem 2.5.

(2) Different from what we have obtained in the single-machine setting, for the resume model, we cannot match the lower bound 5 when $\kappa = 1$ in the multi-machine setting.

Proof. The proof of Theorem E.1 is essentially the same as the proof of single machine setting, in our virtual charging scheme, we charge a completed job in the optimal allocation to some job completed by Γ_2 on the exactly same machine. The difference is that the integral charging scheme for the resume model will not apply to the multiple machines setting any more. We only use the segment charging scheme for the resume model. $\square$

E.2 General Case: $h \geq 1$

To handle this general case, we design a new mechanism Γ_3, which divides the C machines into $\lfloor C/h \rfloor$ equally-sized virtual machines (each consisting of h machines), and treats every virtual machine as a single machine when performing the scheduling. That is, each virtual machine will be used to process one job, and the remaining $C - \lfloor C/h \rfloor \cdot h$ machines will be idle.

Algorithm 6: The allocation rule of Mechanism Γ_3

(1) Divide the C machines into $\lfloor C/h \rfloor$ equal-sized virtual machines.
(2) Run mechanism Γ_2 under the following modification:
  – Capacity: $\lfloor C/h \rfloor$.
  – Demand size: s_j/h for each job j.

As compared to the case of $h = 1$, the setting $h \geq 1$ imposes more flexibilities to the optimal offline allocation. For example, a job may be processed on any number of machines between 1 and h in the optimal allocation and it might not always be executed on exactly h machines. Fortunately, we can use the similar segmental charging idea as $h = 1$ case to resolve the challenge and get the competitive ratio as shown in the following theorem.

Theorem E.2. *Mechanism Γ_3 is IC and has a competitive ratio of $(\frac{4}{(1-\varepsilon)^2} + o(1)) \cdot \frac{\kappa}{\ln \kappa}$ for Γ_3 by setting $\beta = 1 - (1-\varepsilon)^2 \cdot \frac{\ln \kappa}{\kappa}$ for arbitrary small $\varepsilon > 0$, no matter restart model or resume model.*

We have the following discussions for the above theorem. The setting of $h \geq 1$ is more complicated and we could not always obtain the same results as in the setting of $h = 1$. In particular, if h divides C, there will be no idle machine and we may obtain the same competitive ratio as in the setting of $h = 1$. However, when h does not divide C, the idle machines will introduce an additional factor of at most 2 to the competitive ratio. Besides, the competitive ratio for the restart model is no better than that for the resume model, and the competitive ratio cannot reach 5 when $\kappa = 1$.

Proof. Here we only need to show that there exists an optimal allocation (we can view all jobs as resumable jobs in the optimal allocation) such that at any time every job is processed on either exactly h machines or no machine (assuming h divides C). Then we can directly use the results obtained for the special case $h = 1$. Suppose *opt* is an optimal offline allocation and J^* is the set of jobs completed under *opt*. For each $j \in J^*$, we use $m_j(t)$ to denote the number of machines processing j at time t under *opt*. Then we can divide the time into intervals $[t_k, t_{k+1})$, where $k = 0, 1, 2, \cdots$, such that at any time interval $[t_k, t_{k+1})$, $m_j(t)$ does not change for any $j \in J^*$. Now we show how to allocate the jobs at any time interval $[t_k, t_{k+1})$. For the $\lfloor C/h \rfloor$ virtual machines, we allocate the jobs on them one by one, i.e., only if the previous virtual machine is full, we start to allocate jobs on another empty virtual machine from t_k (empty is only with respect to $[t_k, t_{k+1})$). Besides, every job is allocated continuously one by one and the size of allocation is $\int_{t_k}^{t_{k+1}} m_j(t)dt$. It can be easily verified that under this allocation every job $j \in J^*$ is allocated legitimately (j is allocated during $[r_j, d_j]$ and processed by at most h machines at any time) and can complete before its deadline since j is legitimately completed under *opt*. □

References

Azar, Y., Ben-Aroya, N., Devanur, N. R., & Jain, N. (2013). Cloud scheduling with setup cost. In *Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures*, pp. 298–304. ACM.

Bar-Noy, A., Guha, S., Naor, J., & Schieber, B. (2001). Approximating the throughput of multiple machines in real-time scheduling. *SIAM Journal on Computing*, *31*(2), 331–352.

Baruah, S., Koren, G., Mao, D., Mishra, B., Raghunathan, A., Rosier, L., Shasha, D., & Wang, F. (1992). On the competitiveness of on-line real-time task scheduling. *Real-Time Systems*, *4*(2), 125–144.

Baruah, S. K., Haritsa, J., & Sharma, N. (1994). On-line scheduling to maximize task completions. In *Proceedings of Real-Time Systems Symposium*, pp. 228–236. IEEE.

Borodin, A., & El-Yaniv, R. (1998). *Online computation and competitive analysis*, Vol. 2. Cambridge University Press Cambridge.

Chin, F. Y., & Fung, S. P. (2003). Online scheduling with partial job values: Does time-sharing or randomization help?. *Algorithmica*, *37*(3), 149–164.

Ding, J., Ebenlendr, T., Sgall, J., & Zhang, G. (2007). Online scheduling of equal-length jobs on parallel machines. In *Proceedings of the 15th annual European conference on Algorithms*, pp. 427–438. Springer-Verlag.

Ding, J., & Zhang, G. (2006). Online scheduling with hard deadlines on parallel machines. In *Algorithmic Aspects in Information and Management*, pp. 32–42. Springer.

Dürr, C., Jeż, Ł., & Nguyen, K. T. (2012). Online scheduling of bounded length jobs to maximize throughput. *Journal of Scheduling*, *15*(5), 653–664.

Ebenlendr, T., & Sgall, J. (2009). A lower bound for scheduling of unit jobs with immediate decision on parallel machines. In *Approximation and Online Algorithms*, pp. 43–52. Springer-Verlag.

Friedman, E. J., & Parkes, D. C. (2003). Pricing wifi at starbucks: issues in online mechanism design. In *Proceedings of the 4th ACM conference on Electronic commerce*, pp. 240–241. ACM.

Goldman, S. A., Parwatikar, J., & Suri, S. (2000). Online scheduling with hard deadlines. *Journal of Algorithms*, *34*(2), 370–389.

Goldwasser, M. H. (2003). Patience is a virtue: The effect of slack on competitiveness for admission control. *Journal of Scheduling*, *6*(2), 183–211.

Hajek, B. (2001). On the competitiveness of on-line scheduling of unit-length packets with hard deadlines in slotted time. In *Proceedings of the 35th annual Conference on Information Sciences and Systems*.

Hajiaghayi, M., Kleinberg, R., Mahdian, M., & Parkes, D. C. (2005). Online auctions with re-usable goods. In *Proceedings of the 6th ACM conference on Electronic commerce*, pp. 165–174. ACM.

Kolen, A. W., Lenstra, J. K., Papadimitriou, C. H., & Spieksma, F. C. (2007). Interval scheduling: A survey. *Naval Research Logistics (NRL)*, *54*(5), 530–543.

Lavi, R., & Nisan, N. (2004). Competitive analysis of incentive compatible on-line auctions. *Theoretical Computer Science*, *310*, 159–180.

Lavi, R., & Nisan, N. (2015). Online ascending auctions for gradually expiring items. *Journal of Economic Theory*, *156*, 45–76.

Lipton, R. J., & Tomkins, A. (1994). Online interval scheduling. In *In Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, Vol. 94, pp. 302–311.

Lucier, B., Menache, I., Naor, J. S., & Yaniv, J. (2013). Efficient online scheduling for deadline-sensitive jobs. In *Proceedings of the 25th ACM symposium on Parallelism in algorithms and architectures*, pp. 305–314. ACM.

Ma, W., Zheng, B., Qin, T., Tang, P., & Liu, T. (2014). Online mechanism design for cloud computing. *CoRR*, *abs/1403.1896*.

Mashayekhy, L., Nejad, M. M., Grosu, D., & Vasilakos, A. V. (2014). Incentive-compatible online mechanisms for resource provisioning and allocation in clouds. In *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, pp. 312–319. IEEE.

Nguyen, K. T. (2011). Improved online scheduling in maximizing throughput of equal length jobs. In *Computer Science–Theory and Applications*, pp. 429–442. Springer.

Nisan, N. (2007). Introduction to mechanism design (for computer scientists). *Algorithmic game theory*, *209*, 242.

Nisan, N., & Ronen, A. (2001). Algorithmic mechanism design. *Games and Economic Behavior*, *35*, 166–196.

Parkes, D. C. (2007). Online mechanisms. *Algorithmic Game Theory, ed. N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, Cambridge University Press*, 411–439.

Porter, R. (2004). Mechanism design for online real-time scheduling. In *Proceedings of the 5th ACM conference on Electronic commerce*, pp. 61–70. ACM.

Ting, H.-F. (2008). A near optimal scheduler for on-demand data broadcasts. *Theoretical Computer Science*, *401*(1), 77–84.

Wu, X., Gu, Y., Li, G., Tao, J., Chen, J., & Ma, X. (2014). Online mechanism design for VMS allocation in private cloud. In *Network and Parallel Computing*, pp. 234–246. Springer.

Zaman, S., & Grosu, D. (2012). An online mechanism for dynamic vm provisioning and allocation in clouds. In *5th International Conference on Cloud Computing (CLOUD)*, pp. 253–260. IEEE.

Zhang, H., Li, B., Jiang, H., Liu, F., Vasilakos, A. V., & Liu, J. (2013). A framework for truthful online auctions in cloud computing with heterogeneous user demands. In *Proceedings of INFOCOM*, pp. 1510–1518. IEEE.

Zheng, F., Fung, S. P., Chan, W.-T., Chin, F. Y., Poon, C. K., & Wong, P. W. (2006). Improved on-line broadcast scheduling with deadlines. In *Computing and Combinatorics*, pp. 320–329. Springer.