Improved Competitive Analysis of Online Scheduling Deadline-Sensitive Jobs

Patrick Loiseau and Xiaohu Wu
EURECOM, France
{patrick.loiseau,xiaohu.wu}@eurecom.fr

Abstract. We consider the following scheduling problem. There is a single machine and the jobs will arrive for completion online. Each job $j$ is preemptive and, upon its arrival at time $a_j$, its other characteristics are immediately revealed to the machine: the deadline $d_j$, the workload $D_j$ and the value $v_j$. The objective is to maximize the aggregate value of jobs completed by their deadlines. Using the minimum of $\frac{d_j-a_j}{D_j}$ over all jobs as the slackness $s$, a non-committed and a committed online scheduling algorithm $\mathcal{A}$ and $\mathcal{A}_C$ is proposed in [Lucier et al., SPAA’13; Azar et al., EC’15], achieving competitive ratios of $cr_{\mathcal{A}}(s) = 2 + O\left(\frac{1}{\sqrt{3s-1}}\right)$ and $cr_{\mathcal{A}_C}(s) = \frac{cr_{\mathcal{A}}(s \cdot \omega)}{\omega(1-\omega)}$ respectively, where $\omega \in (0, 1)$ and $s \geq \frac{1}{\sqrt{3(1-\omega)}}$.

In this paper, without recourse to the dual fitting technique used in the above works, we propose a simpler and more intuitive analytical framework for $\mathcal{A}$ and $\mathcal{A}_C$, improving $cr_{\mathcal{A}}(s)$ to $1 + O\left(\frac{1}{\sqrt{3s-1}}\right)$ and therefore improving $cr_{\mathcal{A}_C}(s)$ by $\frac{1}{\omega(1-\omega)}$. As stated in [Lucier et al., SPAA’13; Azar et al. EC’15], it is justifiable in scenarios like the online batch processing for cloud computing that $s$ is large, hence the item $O\left(\frac{1}{\sqrt{3s-1}}\right)$ in $cr_{\mathcal{A}}(s)$ can be ignored. Under the above assumption, our analysis brings very significant improvements: from 2 to 1 and from $\frac{1}{\omega(1-\omega)}$ to $\frac{1}{\omega(1-\omega)}$ for $cr_{\mathcal{A}}(s)$ and $cr_{\mathcal{A}_C}(s)$ respectively.

Keywords: Online scheduling, competitive analysis, deadline, value

1 Introduction

1.1 Background and Motivation

In this paper, we reconsider the scheduling problem introduced in [5,6]. In this problem, there is a single machine to process jobs and preemptive jobs arrive online over time. Upon arrival of a job $j$ at time $a_j$, it becomes ready for processing immediately, and its profile specified by a size $D_j$, a deadline $d_j$, and a value $v_j$ is revealed to the machine. The objective is to maximize the aggregate value of jobs completed by their deadlines. For this problem, based on the analytical framework of the dual fitting technique and using the slackness defined as the minimum of $\frac{d_j-a_j}{D_j}$ over all jobs as a metric to measure the performance of an algorithm, Lucier et al. propose and analyze an online
non-committed scheduling algorithm $A$ in [5], achieving a competitive ratio of $cr_A(s) = 2 + \mathcal{O}\left(\frac{1}{\sqrt{s-1}}\right)$. Then, based on the same analytical framework, Azar et al. also proposed a committed scheduling algorithm $A_C$ that achieves a competitive ratio of $cr_{A_C}(s) = \frac{cr_A(s\omega(1-\omega))}{\omega(1-\omega)}$, where $s \geq \frac{1}{\omega(1-\omega)}$ and $\omega \in (0, 1)$.

Here, for committed scheduling, before the execution of a job $j$, the machine will determine whether it will be admitted for execution, and once admitted at a time $t \in [a_j, d_j - D_j]$ and informed of this decision, it will be completed by its deadline definitely; for non-committed scheduling, the machine just seeks for opportunities to execute a job upon its arrival and a job that has been processed partially may not be finished until the deadline. The importance of committed scheduling lies in that, in some scenarios, a job need to know whether it can be finally completed by the deadline before its execution and if not, it can seek for other resources for the completion. The slackness $s$ is used to measure the highest urgency in time of allocating the machine to jobs so that we can ensure the completion of jobs by their deadlines (e.g., in an extreme case, $\frac{d_j - a_j}{D_j} = 1$ means that job $j$ has to be allocated the machine at every time in $[a_j, d_j]$ to ensure its completion by the deadline).

Before the algorithms $A$ and $A_C$, there had been algorithms with competitive ratios polynomial [1] or polylogarithmic [2] in the ratio $\kappa$ between the maximal and minimal values. Here, the algorithm in [2] requires a job to start executing immediately upon arrival to meet the deadline requirement, and as $\kappa$ can be arbitrarily high, these bounds are unrealistic in practice. Constant competitive ratios are only known for special cases, e.g., identical job sizes [3], or job values which are proportional to their sizes [4]; however both of these cases do not encompass realistic settings. A natural goal is to develop constant-factor approximations under assumptions that can be reasonable for realistic input profiles in practice. Hence, Lucier et al. and Azar et al. [5,6] made a slackness assumption that $s$ is large and claimed that this assumption is justifiable in practice especially for the online batch processing scenario in cloud computing where no job extremely pressures the system by requiring immediate and continuous execution so as to meet the deadline. Under such a slackness assumption, the item $\mathcal{O}\left(\frac{1}{\sqrt{s-1}}\right)$ in the above competitive ratio $cr_A(s)$ can be ignored, and $A$ and $A_C$ can therefore be viewed to achieve constant competitive ratios of 2 and $\frac{2}{\omega(1-\omega)}$ that are independent of jobs’ profiles.

In terms of algorithmic analysis, both the algorithms $A$ and $A_C$ in [5,6] are analyzed based on the dual fitting technique. The main power of this technique is to provide an accessible way to bound the optimal social welfare. In this technique, the original problem is first formulated as an integer programming (IP) and the IP is further relaxed as a linear programming (LP). Then, there is an algorithm that can output a feasible solution $Y$ to the original problem, and the technique requires to construct a feasible solution $X$ to the dual of LP. Then, due to the weak duality, the value of the dual under the solution $X$ (that is bounded in the form of the social welfare under the solution $Y$ multiplied by a parameter $\alpha$) will be an upper bound of the optimal value of IP, i.e., the social welfare obtained by an optimal solution to the original problem, where $\alpha \geq 1$. 
Hence, the competitive ratio of the algorithm is $1/\alpha$. Here, this ratio is a lower bound of the ratio of the social welfare obtained by an online algorithm to the social welfare obtained by an optimal offline algorithm with the full knowledge of the characteristics of all jobs in advance.

1.2 Our Results

In this paper, without recourse to the dual fitting technique, we revisit the analysis in [5,6] and propose a new framework to analyze the online algorithms $A$ and $A_C$, that yields improved competitive ratios. More specifically, our contributions are summarized as follows:

1. We propose a simpler and more intuitive analytical framework for the algorithm $A$ in [5], also improving the competitive ratio $cr_A(s)$ of $A$ from $2 + \mathcal{O}\left(\frac{1}{\sqrt{s-1}}\right)$ to $1 + \mathcal{O}\left(\frac{1}{\sqrt{s-1}}\right)$. In this analysis, we extend an important stretching lemma (i.e., Lemma 2) using the new proof technique we develop. The lemma also plays a core role in the design of the algorithm $A_C$ in [6].

2. The proposed analysis of $A$ directly improves the competitive ratio $cr_{A_C}(s) = \frac{cr_A(s\omega(1-\omega))}{\omega(1-\omega)}$ of $A_C$ by $\frac{1}{\omega(1-\omega)}$, where $\omega \in (0, 1)$ and $s \geq \frac{1}{\omega(1-\omega)}$. We also propose a more intuitive framework to analyze $A_C$ using the extended stretching lemma.

3. Under the slackness assumption in [5,6] that $s$ is large, we can therefore ignore the term $\mathcal{O}\left(\frac{1}{\sqrt{s-1}}\right)$ in the competitive ratios, and our analysis improves the competitive ratios of $A$ and $A_C$ from 2 to 1, and from $\frac{2}{\omega(1-\omega)}$ to $\frac{1}{\omega(1-\omega)}$ respectively. Hence, under the slackness assumption, our analysis shows the optimality of the algorithm $A$.

Finally, it is worth noting that both of the algorithms $A$ and $A_C$ are important for the multiple machines case of the problem considered in this paper to which they have been successfully extended in [5,6]. Our analysis therefore has the potential to bring similar improvements to the multiple machine case as well.

2 Model and Problem Description

There is one machine and job requests are submitted to the machine over time. The machine is fully available throughout time and is managed by a scheduler, which determines the resource allocation. The input is a finite set of batch jobs, denoted $\mathcal{J}$. These jobs arrive to the system online, over the (continuous) time interval $\mathcal{R}^+ = [0, +\infty)$. Every job $j \in \mathcal{J}$ is revealed to the system only upon its arrival time $a_j$. Upon arrival, each job $j$ specifies its deadline, size and value. The deadline $d_j$ indicates the latest acceptable completion time for job $j$. The interval $W_j = [a_j, d_j]$ is called the availability window of job $j$. The size $D_j$ of job $j$ is the total amount of resource required to complete the job (e.g., in CPU hours). A value $v_j$ is gained by the system if and only if job $j$ is fully
executed by its deadline (i.e., allocated $D_j$ units of resource by time $d_j$). Partial execution of a job does not yield any value.

For any set of jobs $S \subseteq J$, we denote by $v(S) = \sum_{j \in S} v_j$ its aggregate value. We denote the ratio of the value of a job $j$ and its demand, i.e., its value density, by $\rho_j = v_j/D_j$. The goal of the scheduler is to maximize the aggregate value of jobs fully completed by their deadlines. The scheduler is not required to complete all jobs. Specifically, if a job reaches its deadline without being completed, it will not gain any benefit to further allocate the resource to it. In addition, jobs can be preempted and resumed later without any additional cost.

The performance guarantees of the online algorithms in this paper are measured by the parameter $s (\geq 1)$ called the slackness of the input. Here, we say that the input has slackness $s$ if for each job $j$, $d_j - a_j \geq s \cdot D_j$. The slackness parameter $s$ limits the tightness of a job's deadline with respect to its size.

2.1 Notation

For an online algorithm $A$ and an input sequence of jobs $J$, denote by $A(J)$ the set of jobs that are fully completed before their deadlines over an online sequence of arriving jobs $J$. The throughput gained by $A$ is $v(A(J))$. Let $OPT(J)$ denote the set of jobs completed by an optimal offline scheduling algorithm in which full knowledge of the characteristics of all jobs $J$ is available in advance. The standard notion of competitive ratio is used to measure the worst-case performance guarantee of an online algorithm, defined as follows:

$$\text{cr}_A = \max_{J} \left\{ \frac{v(OPT(J))}{v(A(J))} \right\}.$$ 

To analyze and design algorithms, the following notation is further introduced. Denote by $j_A(t)$ the job being processed by the machine at time $t$ and $\rho_A(t)$ the value-density at this moment. Set $y_j(t) = 1$ if $j_A(t) = j$ and $y_j(t) = 0$ otherwise. A job $j$ is fully completed by the deadline if and only if $\int_{a_j}^{d_j} y_j(t) \, dt \geq D_j$. The starting point $st(y_j) = \min \{ \{ t | y_j(t) = 1 \} \cup \{ +\infty \} \}$ of job $j$ is the earliest time at which $j$ begins its execution. For a given $\mu \in \mathbb{R}^+$, denote by $W_{-\mu}$ the time interval $[a_j, d_j - \mu D_j]$, and by $A_{-\mu}(t) = \{ j \in J | t \in W_{-\mu} \}$ the set of jobs for which the available time remaining at $t$ is no less than $\mu D_j$. The algorithms in this paper will impose that the starting time of a job $j$ cannot be later than $d_j - \mu D_j$. Upon completion of the algorithm $A$, the final jobs are divided into three classes: (1) jobs $J^F$ fully processed (i.e., completed) by their deadlines; (2) jobs $J^P$ that were partially processed and have begun their execution but were not completed on time; and (3) unprocessed jobs $J^E$ that have never begun their execution.

3 Analysis of Non-Committed Algorithm

In this section, we propose a new analysis for the non-committed scheduling algorithm $A$ in [5] without recourse to dual fitting technique on which the analysis of $A$ in [5] builds. Our analysis is somewhat simpler and more intuitive and
Algorithm 1: Non-Committed Scheduling

\[
\forall t, J^P(t) = \{ j \in J \mid j \text{ partially processed by } A_T \text{ at time } t \land t \in [a_j, d_j]\}.
\]

\[J^E(t) = \{ j \in J \mid j \text{ unprocessed by } A_T \text{ at time } t \land t \in [a_j, d_j - \mu D_j]\}.\]

**Event 1:** On arrival of job \(j\) at time \(t = a_j\):

1. Call ClassPreemptionRule\((t)\).

**Event 2:** Either \(j\) is fully completed at time \(t \leq d_j\) or is partially processed at \(t = d_j\):

1. Resume execution of job \(j' = \arg \max \{ \rho_{j'} \mid j' \in J^P(t)\}\).
2. Call ClassPreemptionRule\((t)\).

**ClassPreemptionRule\((t)\):**

1. \(j \leftarrow \text{job currently being processed.}\)
2. \(j^* \leftarrow \arg \max \{ \rho_{j^*} \mid j^* \in J^E(t)\}\).
3. If \((j^* \succ j)\):
   (a) Preempt \(j\) and execute \(j^*\).

It improves the competitive ratio of \(A\) from \(2 + \Theta\left(\frac{1}{\sqrt{\gamma - 1}}\right)\) to \(1 + \Theta\left(\frac{1}{(\sqrt{\gamma - 1})^2}\right)\). It also enables us to propose an improved analysis for the algorithm in [6] in the next section.

3.1 Non-Committed Scheduling Algorithm

We present the algorithm \(A\) from [5] as Algorithm 1. Let \(\gamma \in (1, +\infty)\) and \(\mu \in [1, s]\) be parameters to be fixed. The underlying principles of \(A\) consist of the following two policies:

**Policy 1** A pending job \(j'\) can preempt a running job \(j\) only if \(\rho_{j'} > \gamma \rho_j\).

**Policy 2** A job \(j\) cannot begin its execution after time \(d_j - \mu D_j\).

The following lemma is directly from [3]:

**Lemma 1.** The algorithm \(A\) holds the following properties for the executed jobs:

1. Any allocated job \(j \notin J^E\) satisfies \(st(y_j) \leq d_j - \mu D_j\).
2. For any \(t\), let \(j' \in J^P\) be a job such that either it has been partially processed at time \(t\) by \(A\), or \(j' \in A^-\mu(t)\); and let \(j\) be a job running at time \(t\). Then, \(\rho_{j'} \leq \gamma \rho_j\).
3. Let \(j' \in J^P\) be a job that has been partially processed at time \(t\) by \(A\). Any job \(j\) running at \(t\) such that \(st(y_{j'}) \leq t \leq d_{j'}\) satisfies \(st(y_{j'}) \leq st(y_j)\).
3.2 Analysis

The algorithmic analysis consists of three steps: (1) we give an upper bound $U_1(J)$ of the optimal social welfare using all the three types of jobs $J^E$, $J^P$, and $J^F$; (2) we study how to bound $U_1(J)$ using the processed jobs $J^P$ and $J^F$ and give such a bound $U_2(J^F \cup J^P)$; and (3) we study how to bound $U_2(J^F \cup J^P)$ using the jobs $J^F$ fully completed by their deadlines. Finally, we obtain the upper bound of $U_1(J)$ in the form of $\alpha \cdot v(J_F)$, where $\alpha \geq 1$, and $\alpha$ is the competitive ratio of $A$. In this analysis, we will observe how the scheduling policies in $A$ affects the bounds in the above three steps.

Step 1. Let $\rho(t) = \max\{\rho_j \mid t \in [a_j, d_j]\}$ and we have that

**Proposition 1.** $\int_0^\infty \rho(t)dt$ is an upper bound of the optimal social welfare for the problem considered in this paper.

**Proof.** At any time $t \in R^+$, $\rho(t)$ is the maximal value density of a job that can be executed at this moment and the proposition therefore holds. □

Proposition 1 completes the first step of our analysis. In fact, the dual fitting technique is used in [5,6] as a tool to find the upper bound of the optimal social welfare. Our bound here presents a simpler and different way than dual fitting technique to bound the optimal social welfare, and, as we will see in Step 2, it also reduces the upper bound of Theorem 3.4 in [5] by $v(J_F)$ through extending some conclusions in [5]. This finally leads us to improve the competitive ratio of $A$ by 1.

Step 2. We now proceed to the second step of our analysis. Let $\rho^\mu(t) = \max\{\rho_j \mid j \in A^\mu(t)\}$ and we first give an extended version of the stretching lemma in [5]. Here, the stretching lemma holds with regard to all three types of jobs $J^E \cup J^P \cup J^F$ while it is shown to hold only with regard to $J^E \cup J^P$ in [5]. The specific technique to prove the stronger conclusion in the proof of the extended stretching lemma (i.e., Lemma 2 below) is different from the one in the full version of [5]. The stronger conclusion in the proof of Lemma 2 is also important in the analysis of the committed scheduling algorithm in the next section.

**Lemma 2.** $\int_0^\infty \rho(t)dt \leq \frac{s}{s-\mu} \int_0^\infty \rho^\mu(t)dt$.

**Proof.** Consider all the jobs in $J$ in the non-increasing order of value-densities and assume without loss of generality that $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_n$. For every job $i$, let $\tau_i = a_i + \frac{\rho_i}{\mu}(d_i-a_i)$ and $\beta^\mu(t) = \max\{\rho_j \mid t \in [\bar{a}_j, \tau_i]\}$, where $\tau_i \leq d_i - \mu D_i$.

Here, we will prove a stronger conclusion than Lemma 2 that is simpler to prove:

$\int_0^\infty \rho(t)dt \leq \frac{s}{s-\mu} \int_0^\infty \beta^\mu(t)dt$.

We define the following two sets:

$W(j) = \bigcup_{j'=1}^j [a_{j'}, d_{j'}]; \quad W^\mu(j) = \bigcup_{j'=1}^j [a_{j'}, \tau_{j'}]$. 
In particular, we define $W_{(0)} = W_0^{-\rho} = \emptyset$.

**Extended Interval.** We also define a set $\tilde{W}_{(j)}$ that is an extension of $W_{(j)}^{-\rho}$ as follows: the set $W_{(j)}^{-\rho}$ can be viewed as an union of several disjoint continuous intervals: $[x_1, y_1], \ldots, [x_L, y_L]$, where $0 \leq x_1 < y_1 < x_2 < \cdots < y_L$. For ease of exposition, let $x_{L+1} = +\infty$. Initially, let $\Delta = 0$ and, for every $1 \leq i \leq L$, we continuously extend each interval $[x_i, y_i]$ to $[x_i, y_i']$, where $y_i' = \min\{x_{i+1}, x_i + \frac{s}{\beta}(y_i - x_i)\}$. If $y_i' < x_{i+1}$, let $y_i'' = \min\{y_i' + \Delta, x_{i+1}\}$, update $\Delta$ to be $\Delta - (y_i'' - y_i')$, and let $y_i' = y_i''$; otherwise, update $\Delta$ to be $\Delta + \max\{0, x_i + \frac{s}{\beta} \cdot (y_i - x_i) - x_{i+1}\}$. Note that when the extension of the last interval is completed, $\Delta = 0$ and $|\tilde{W}_{(j)}| = \frac{s}{\beta} |W_{(j)}^{-\rho}|$. In particular, we set $\tilde{W}_{(0)} = \emptyset$.

In order to prove our stronger conclusion, we need the following two claims:

**Claim 1.** For every $0 \leq j \leq n - 1$, if we remove the first $j$ jobs, we have

$$\frac{s}{s - \rho} \left| [a_{j+1}, \tau_{j+1}] - W_{(j)}^{-\rho} \right| = \left| \tilde{W}_{(j+1)} - \tilde{W}_{(j)} \right|.$$  

(1)

**Claim 2.** In terms of the structure of $\int_{W_n} \rho_{\max}(t)dt$ on value densities, we have the following conclusion:

$$\int_{W_n} \rho_{\max}(t)dt \leq \rho_1 \left| \tilde{W}_1 - \tilde{W}_0 \right| + \cdots + \rho_n \left| \tilde{W}_n - \tilde{W}_{n-1} \right|.$$  

(2)

Then, combining Equality (1) and Inequality (2) completes the proof of Lemma 2 since

$$\int_{W_n} \rho_{\max}(t)dt \leq \sum_{i=1}^{n} \frac{s}{s - \rho} \cdot \rho_i \cdot \left| [a_i, \tau_i] - W_{(i-1)}^{-\rho} \right| = \frac{s}{s - \rho} \sum_{i=1}^{n} \beta^{-\rho}_{\max}(t)dt.$$

From the way that we define $\tilde{W}_j$ ($1 \leq j \leq n$), we have that $\left| \tilde{W}_{(j+1)} - \tilde{W}_{(j)} \right| = \frac{s}{s - \rho} \left| W_{(j+1)}^{-\rho} - W_{(j)}^{-\rho} \right|$. Then, Claim 1 follows since $W_{(j+1)}^{-\rho} - W_{(j)}^{-\rho} = [a_{j+1}, \tau_{j+1}] - W_{(j)}^{-\rho}$. To prove Claim 2, we need the following claim whose proof is given in the appendix.

**Claim 3.** From the way that we construct $W_{(j)}$ and $\tilde{W}_{(j)}$, we have $W_{(j)} \subseteq W_{(j+1)}$, $\tilde{W}_{(j)} \subseteq \tilde{W}_{(j+1)}$, and $W_{(j)} \subseteq \tilde{W}_{(j)}$.

Once we have the inclusion relations among sets, we can associate every $t \in W_j - \tilde{W}_{j-1}$ with a value $\beta(t) = \rho_j$. Then, it is obvious that there exist sets $W_1' \subseteq W_1, W_2' \subseteq W_2 - W_1', \ldots, W_n' \subseteq W_n - \sum_{j'=1}^{n-1} W_{j'}$, so that the following properties are satisfied: (1) $|W_i'| = |W_i - W_{i-1}|$, and (2) for every $t \in W_i'$, $\beta(t)$ is no less than $\rho_i$ ($1 \leq i \leq n$). Hence, according to the way that we define $\rho_{\max}(t)$
and $\beta_{\max}^{-\mu}(t)$, we have that

$$\int_{W_n} \rho_{\max}(t) dt = \rho_1 |[a_1, d_1] - W_0| + \cdots + \rho_n |[a_n, d_n] - W_{n-1}|$$

$$= \rho_1 |W_1| + \rho_2 |W_2 - W_1| + \cdots + \rho_n |W_n - W_{n-1}|$$

$$\leq \int_{W_1} \beta(t) dt + \int_{W_2} \beta(t) dt + \cdots + \int_{W_n} \beta(t) dt$$

$$\leq \rho_1 |\widetilde{W}_1 - \widetilde{W}_0| + \rho_2 |\widetilde{W}_2 - \widetilde{W}_1| + \cdots + \rho_n |\widetilde{W}_n - \widetilde{W}_{n-1}| . \quad \square$$

Recall that $\rho_A(t)$ is the value density of the job processed by $A$ at time $t$. The lemma below follows from Lemma 1.(2) directly:

**Lemma 3.** During the execution of $A$, we always have that

$$\rho_{\max}^{-\mu}(t) \leq \gamma \cdot \rho_A(t).$$

The proposition below follows from Lemmas 2 and 3 directly:

**Proposition 2.** $\int_0^\infty \rho_{\max}(t) dt \leq \gamma \cdot \frac{s}{s-\mu} \cdot \int_0^\infty \rho_A(t) dt.$

**Step 3.** Lucier et al. showed the way to bound $U_2(\mathcal{J}^F \cup \mathcal{J}^P)$ through the jobs $\mathcal{J}^F$ completed by their deadlines in [5], and we do not repeat their analysis in this paper. Hence, in the third step, we adopt the conclusion in [5] directly:

**Proposition 3.** $\int_0^\infty \rho_A(t) dt \leq v(\mathcal{J}^F) \cdot \left[ \frac{(\gamma-1)(\mu-1)}{(\gamma-1)(\mu-1)-1} \right].$

**Competitive Ratio.** Finally, the following theorem follows directly from the above Propositions 1, 2, and 3

**Theorem 1.** The competitive ratio of the algorithm $A$ is at most

$$\gamma \cdot \frac{s}{s-\mu} \cdot \left[ \frac{(\gamma-1)(\mu-1)}{(\gamma-1)(\mu-1)-1} \right].$$

Under the slackness assumption where the slackness $s$ is large in [56], when $\gamma = \frac{\sqrt{\mu}}{\sqrt{\mu} - 1}$ and $\mu = \frac{s^2}{3}$, we obtain the following bound for the competitive ratio of $A$:

$$\text{cr}_A(s) \leq 1 + \mathcal{O}\left( \frac{1}{(\sqrt{s} - 1)^2} \right). \quad (3)$$

The bound above is better than the bound $\frac{1}{2} + \mathcal{O}\left( \frac{1}{(\sqrt{s} - 1)^2} \right)$ in [57]. Under the slackness assumption, we can therefore ignore the item $\mathcal{O}\left( \frac{1}{(\sqrt{s} - 1)^2} \right)$ in the competitive ratios and improve the competitive ratio of $A$ from 2 to 1.

\textsuperscript{1} In this paper, we adopt the most recent bound for the competitive ratio of the algorithm $A$ which is given in the conclusion of [7].
4 Committed Scheduling Algorithm

In this section, based on the analysis proposed in Section 3, we propose a simpler analysis of the committed scheduling algorithm $\mathcal{A}_C$ in [6] without recourse to dual fitting technique on which the analysis of $\mathcal{A}_C$ in [6] builds. The analysis in Section 3 also enables improving the competitive ratio of $\mathcal{A}_C$ by $\frac{1}{\omega(1-\omega)}$, where $\omega \in (0, 1)$ and $s \geq \frac{1}{\omega(1-\omega)}$.

The committed scheduling algorithm $\mathcal{A}_C$ proposed in [6] consists of two components: (1) the simulator: assume that there is an imaginary server that is used to simulate the execution of the non-committed algorithm $\mathcal{A}$ to decide whether an arriving job will be sent to the real server for full execution; and (2) the server: the real server will fully process the jobs chosen by the simulator.

Simulator. The simulator runs an online non-committed scheduling algorithm $\mathcal{A}$. Every arriving job $j$ is automatically sent to the simulator with a virtual type $\tau_j^v = (v_j, D_j^v, a_j, d_j^v)$, where $d_j^v = d_j - \omega(d_j - a_j)$ is the virtual deadline of $j$, and $D_j^v = D_j / \omega$ is the virtual demand of $j$. If $\mathcal{A}$ completes the virtual request of job $j$ by its virtual deadline, then $j$ is admitted and sent to the server.

Server. The server receives the admitted jobs once they have been completed by the simulator, and begins to really processes them according to the Earliest Deadline First (EDF) allocation rule. That is, at any time $t$ the server processes the job with the earliest deadline out of all the admitted jobs sent here by the simulator.

Here, in order to ensure the job $j$ submitted to the server to be completed by the deadline, the following condition is required: $s \geq \frac{1}{\omega(1-\omega)}$. The detailed proof that the server can produce a feasible schedule for the jobs chosen by the simulator can be found in [6] and we will not repeat it.

Let $\rho^\text{max}(t) = \{\rho_j | t \in [a_j, d_j]\}$, and $\rho^\text{max}(t) = \{\frac{\rho_j}{D_j^v} | t \in [a_j, d_j]\}$. Then, we have that $\rho^\text{max}(t) = \omega \cdot \rho^\text{max}(t)$. Let $\rho^\text{max}(t) = \{\rho_j | t \in [a_j, d_j]\}$. Using the stronger conclusion in the proof of the extended stretching lemma (i.e., Lemma 2), we have that

Proposition 4. $\int_0^\infty \rho^\text{max}(t)dt \leq \frac{1}{1-\omega} \int_0^\infty \rho^\text{max}(t)dt$.

Let $\mathcal{J}_C^F$ denote the set of virtual jobs fully completed by their virtual deadlines. Since $\frac{d_j^v-a_j}{D_j^v} = \frac{(1-\omega)(d_j-a_j)}{D_j}$, the slackness of all virtual jobs is $s \cdot \omega(1-\omega)$. Then, according to our analysis in Section 3 we have $\int_0^\infty \rho^\text{max}(t)dt = \frac{1}{s} \cdot \int_0^\infty \rho^\text{max}(t)dt \leq \frac{1}{s} \cdot \omega(\mathcal{J}_C^F) \cdot cr_{\mathcal{A}}(s \cdot \omega(1-\omega))$. Here, $\omega(\mathcal{J}_C^F)$ is also the total value of jobs completed by the server. Hence, by Propositions 1 and 4 we have the following conclusion:

Theorem 2. The above committed scheduling algorithm $\mathcal{A}_C$ obtains a competitive ratio of

$$cr_{\mathcal{A}_C}(s) \leq \frac{cr_{\mathcal{A}}(s \cdot \omega(1-\omega))}{\omega(1-\omega)}, \omega \in (0, 1) \text{ and } s \geq \frac{1}{\omega(1-\omega)}.$$
Hence, by Theorems 1 and 2, our analysis in fact shows that $A_C$ achieves a competitive ratio

$$\frac{1}{\omega(1-\omega)} \cdot \left[1 + O\left(\frac{1}{\sqrt{s \omega(1-\omega)-1}}\right)^2\right].$$

In contrast, the competitive ratio of $A_C$ obtained in the dual fitting based analysis of [6] is

$$\frac{1}{\omega(1-\omega)} \cdot \left[2 + O\left(\frac{1}{\sqrt{s \omega(1-\omega)-1}}\right)^2\right].$$

Further, under the slackness assumption where the item $O\left(\frac{1}{\sqrt{s \omega(1-\omega)-1}}\right)^2$ can be ignored, our analysis improves the competitive ratio of $A_C$ from $\frac{2}{\omega(1-\omega)}$ to $\frac{1}{\omega(1-\omega)}$.

5 Conclusion

In this paper, we revisit the previous dual fitting based analysis of two algorithms $A$ and $A_C$ for online scheduling of deadline-sensitive jobs on a single machine, and propose a simpler analytical framework for this type of problems without recourse to the dual fitting technique. As a result, the proposed analysis improves the competitive ratio $cr_A(s)$ of $A$ from $2 + O\left(\frac{1}{\sqrt{s \omega(1-\omega)-1}}\right)$ to $1 + O\left(\frac{1}{\sqrt{s \omega(1-\omega)-1}}\right)$, and therefore improves the competitive ratio $cr_{A_C}(s) = \frac{cr_A(s \cdot \omega(1-\omega))}{\omega(1-\omega)}$ of $A_C$ by $\frac{1}{\omega(1-\omega)}$, where the slackness $s$ is the minimum of the ratios of deadline minus arrival time to workload, $\omega \in (0, 1)$, and $s \geq \frac{1}{\omega(1-\omega)}$. Under the slackness assumption that $s$ is large that is justifiable in scenarios like scheduling online batch jobs in cloud computing, the item $O\left(\frac{1}{\sqrt{s \omega(1-\omega)-1}}\right)^2$ in the above competitive ratios can be ignored, and our analysis therefore improves the competitive ratios from 2 to 1 and from $\frac{2}{\omega(1-\omega)}$ to $\frac{1}{\omega(1-\omega)}$ for $A$ and $A_C$ respectively. The algorithms $A$ and $A_C$ are also important for the multiple machines case of the problem considered in this paper since they have been extended there. As a future work, one may consider extending the proposed analytical framework to the multiple machines case of the problem of this paper.

References

1. Gilad Koren, and Dennis Shasha. "D”ove: an optimal on-line scheduling algorithm for overloaded real-time systems." In Real-Time Systems Symposium, 1992, pp. 290-299. IEEE, 1992.
2. Ran Canetti, and Sandy Irani. "Bounding the power of preemption in randomized scheduling." SIAM Journal on Computing 27, no. 4 (1998): 993-1015.
3. Mohammad Hajigahey. "Online auctions with re-usable goods." In Proceedings of the 6th ACM conference on Electronic commerce, pp. 165-174. ACM, 2005.
4. Amotz Bar-Noy, Ran Canetti, Shay Kutten, Yishay Mansour, and Baruch Schieber. "Bandwidth allocation with preemption." In Proceedings of the twenty-seventh annual ACM symposium on Theory of computing, pp. 616-625. ACM, 1995.
5. Brendan Lucier, Ishai Menache, Joseph Seffi Naor, and Jonathan Yaniv. "Efficient online scheduling for deadline-sensitive jobs." In Proceedings of the 25th ACM symposium on Parallelism in algorithms and architectures, pp. 305-314. ACM, 2013.
6. Yossi Azar, Inna Kalp-Shaltiel, Brendan Lucier, Ishai Menache, Joseph Naor, and Jonathan Yaniv. "Truthful Online Scheduling with Commitments." In Proceedings of the 16th ACM conference on Electronic commerce. ACM, 2015.
7. Navendu Jain, Ishai Menache, Joseph Naor, and Jonathan Yaniv. "Near-optimal scheduling mechanisms for deadline-sensitive jobs in large computing clusters." ACM Transactions on Parallel Computing, Vol. 2 Issue 1, Article 3, 29 pages, 2015.

A Appendix: Proofs

Proof (of Claim 3). From the way that we construct \( W(j) \) and \( W(j)^-\mu \), we have \( W(j) \subseteq W(j+1) \) and \( W(j)^-\mu \subseteq W(j+1)^-\mu \). Denote by \( W(j)^-\mu = [x,i,y,i] \cup \cdots \cup [x,i,y,i] \) (\( i = j \) or \( j+1 \)), where \( 0 \leq x,i < y,i < x,i+2 \cdots < y,i \). To show that \( W(j) \subseteq W(j+1) \), we separate two cases for discussion:

1. \( [a,j+1,\tau(j+1)] \cap W(j)^-\mu = \emptyset \). In this case, \( L_j+1 = L_j+1 + 1 \). Then, for ease of exposition, let \( [x,j,0,y,j,0] = [x,j,0+1,y,j,0+1] = [x,j+1,\tau(j+1),y,j+1,\tau(j+1)+1] \) (\( j = j+1 \)) and \( [x,j,L_j+1,1,y_j,L_j+1+1] = [M, M+1, M+1+1] \), where \( M \) is large enough. Then, the extension of \( [x,j,0,y,j,0] \) and \( [x,j,L_j+1,1,y_j,L_j+1+1] \) (resp. \( [x,j+1,0,y,j+1,0] \) and \( [x,j+1,L_j+1+1,1,y_j+1,L_j+1+1+1] \)) will not intersect with \( \tilde{W}_j \) (resp. \( \tilde{W}_j+1 \)). Then, there must exist an integer \( i' \) (\( 0 \leq i' \leq L_j \)) such that (1) \( y,i' < a,j+1 < \tau(j+1) \leq x,i' \), (2) \( [a,j+1,\tau(j+1)] = [x,j+1,i'+1,y,j+1,i'+1+1] \), \( [x,j+1,i'+1,y,j+1,i'+1+1] \), \( 0 \leq i' \leq i' \) and \( [x,j+1,i'+1,y,j+1,i'+1+1] \), \( 0 \leq i' \leq i' \leq L_j+1+1 \). In the two processes of extending \( W(j)^-\mu \) and \( W(j+1)^-\mu \) to generate \( W(j) \) and \( \tilde{W}(j+1) \), we have that \( [x,j,i',y,j+1,i'] \subseteq [x,j+1,i,y,j+1,i] \) (\( 0 \leq i' \leq i' - 1 \)), and after completing the extension of \( [x,j+1,i',y,j+1,i'] \) and \( [x,j+1,i',y,j+1,i'] \), the current \( \Delta \) in two processes are the same. Then, due to the effect of \([a,j+1,\tau(j+1)]\), after completing the extension of \( [x,j,i',y,j,i'] \) and \( [x,j+1,i',y,j+1,i'] \), the \( \Delta \) in the former is always no more than the \( \Delta \) in the latter and \( [x,j,i',y,j,i'] \subseteq [x,j+1,i',y,j+1,i'] \). Hence, after completing the extension of \( [x,j,i',y,j,i'] \) and \( [x,j,i',y,j,i'] \), \( [x,j+i+1] \subseteq [x,j+i+1] \) (\( i' + 1 \leq i \leq L_j \)). Finally, we have that \( W(j) \subseteq W(j+1) \).

2. \( [a,j+1,\tau(j+1)] \cap W(j)^-\mu \neq \emptyset \). In this case, there exists an interval \( [x,j+1,i,y,j+1,i] \) such that (1) \( [x,j+1,0,y,j+1,0] \cup \cdots \cup [x,j+1,i,y,j+1,i] \subseteq [x,j+1,i,y,j+1,i] \), (2) \( [x,j,i,y,j,i] = [x,j+1,i,y,j+1,i] \) (\( 1 \leq i \leq i+1 \)), and (3) \( [x,j+1,i,y,j+1,i] = [x,j+1,i-1,y,j+1,i] \) (\( i+2 \leq i \leq L_j \)), where \( 1 \leq i+1 \leq L_j+1 \), and \( 1 \leq i+2 \leq L_j+2 \). In the two processes of extending \( W(j)^-\mu \) and \( W(j+1)^-\mu \) to generate \( W(j) \) and \( \tilde{W}(j+1) \), we have that \( \tilde{W}(j) \subseteq \tilde{W}(j+1) \).
\( \overline{\mathcal{W}}(j+1) \), \([x_{j,i}, y'_{j,i}]\) for all \( 0 \leq i \leq i_1 - 2 \) if \( i_1 \geq 2 \) or for \( i = 0 \) if \( i_1 = 1 \). After completing the extension of \([x_{j,i_1-1}, y_{j,i_1-1}]\) and \([x_{j+1,i_1-1}, y_{j+1,i_1-1}]\), (1) in the case where \( i_1 \geq 2 \) and \( y'_{j+1,i_1-1} < x_{j+1,i_1} \), we have \([x'_{j,i_1-1}, y'_{j,i_1-1}]\) = \([x_{j+1,i_1-1}, y_{j+1,i_1-1}]\) and \( \Delta = 0 \) in both the extension processes; (2) in the case where \( i_1 \geq 2 \) and \( y'_{j+1,i_1-1} = x_{j+1,i_1} \), the \( \Delta \) in the former is no more than the \( \Delta \) in the latter; (3) in the case where \( i_1 = 1 \), we have \( \Delta = 0 \) in both the extension processes at this moment. At this moment, record the current \( \Delta \) in the former and latter respectively as \( \Delta_{j,i_1-1} \) and \( \Delta_{j+1,i_1-1} \). Subsequently, if \( i_1 < i_2 \), observe the process of extending \( [x_{j,i_1}, y_{j,i_1}] \), \( \cdots \), \([x_{j,i_2-1}, y_{j,i_2-1}]\) and after completing these extensions, the current \( \Delta \) becomes \( \Delta_{j,i_1-1} + \frac{\mu}{s-\mu} \sum_{i=1}^{i_2-1} (y_{j,i} - x_{j,i}) - \sum_{i=1}^{i_2-1} (y'_{j,i} - y_{j,i}) \) and record it as \( \Delta_{j,i_2-1} \); if \( i_1 = i_2 \), let \( \Delta_{j,i_2-1} = \Delta_{j,i_1-1} \). Then, we have that \( \Delta_{j,i_2-1} + \frac{\mu}{s-\mu} \cdot (y_{j,i_2} - x_{j,i_2}) \leq \Delta_{j+1,i_1-1} + \frac{\mu}{s-\mu} \cdot (y_{j+1,i_1} - x_{j+1,i_1}) \). Now, observe the extension of \([x_{j,i_2}, y_{j,i_2}]\) and \([x_{j+1,i_1}, y_{j+1,i_1}]\), \([x_{j,i_2}, y_{j,i_2}]\) can be extended by at most \( \min \{ \Delta_{j,i_2-1} + \frac{\mu}{s-\mu} \cdot (y_{j,i_2} - x_{j,i_2}), x_{j+1,i_1} - y_{j,i_2} \} \) from the time \( y_{j,i_2} \) towards the later time, and \([x_{j+1,i_1}, y_{j+1,i_1}]\) can be extended by at most \( \min \{ \Delta_{j+1,i_1-1} + \frac{\mu}{s-\mu} \cdot (y_{j+1,i_1} - x_{j+1,i_1}), x_{j+1,i_1} - y_{j,i_2} \} \) from the time \( y_{j+1,i_1} \) towards the later time. Since \( y_{j,i_2} \leq y_{j+1,i_1} \), we have that \( y'_{j,i_2} \leq y_{j+1,i_1} \). Hence, after completing the extension of \([x_{j,i_2}, y_{j,i_2}]\) and \([x_{j+1,i_1}, y_{j+1,i_1}]\), we have that \( \cup_{i=1}^{i_2} [x_{j,i}, y'_{j,i}] \subseteq \cup_{i=1}^{i_2} [x_{j,i}, y_{j,i}] \), and the \( \Delta \) in the former is still no less than the \( \Delta \) in the latter; further, we can easily derive that \( [x_{j,i}, y'_{j,i}] \) \( \subseteq [x_{j+1,i_2+1}, y_{j+1,i_2+1}] \) for all \( i_2 + 1 \leq i \leq L_j \). Finally, we have that \( \overline{\mathcal{W}}(j) \subseteq \overline{\mathcal{W}}(j+1) \).

In the next, we prove \( \mathcal{W}(j) \subseteq \overline{\mathcal{W}}(j) \). We only need to prove that, it holds that \([a_i, d_i] \subseteq \overline{\mathcal{W}}(j)\) for every \( 1 \leq i \leq j \). Let \([a_i, \tau_i] \subseteq [x_{j,i_3}, y_{j,i_3}]\) for some \( 1 \leq i_3 \leq L_j \). For some \( t \in [\tau_i, d_i] \), let \( t \in [x_{j,i_4}, x_{j,i_4+1}] \) for some \( i_4 \leq i_3 \leq L_j \). Then, by constraint, we only need to show that it does not hold that \( y'_{j,i_4} < x_{j,i_4+1} \) and \( t \in (y'_{j,i_4}, x_{j,i_4+1}) \). If it holds, we have that after completing the extension of \([x_{j,i_4}, y_{j,i_4}]\) the current \( \Delta = 0 \), it means that from the time \( y_{j,i_3} \) toward the later time, the time interval has been continuously extended by at least \( \frac{\mu}{s-\mu} \sum_{i=i_3}^{i_2} (y_{j,i} - x_{j,i}) \), which contradicts with \( t - y_{j,i_3} \leq \tau_i \leq \frac{\mu}{s-\mu} \cdot (\tau_i - a_i) \leq \frac{\mu}{s-\mu} \cdot (y_{j,i} - x_{j,i}) \). \( \square \)