Online Competitive Algorithms for Maximizing Weighted Throughput of Unit Jobs

Yair Bartal* Francis Y. L. Chin† Marek Chrobak‡¶
Stanley P. Y. Fung† Wojciech Jawor‡¶ Ron Lavi*
Jiří Sgall§¶ Tomáš Tichý§¶

Abstract. We study an online buffer management problem for networks supporting Quality-of-Service (QoS) applications. Packets with different QoS values arrive at a network switch and are to be sent along an outgoing link. Due to overloading conditions, some packets have to be dropped. The objective is to maximize the total value of packets that are sent. We formulate this as an online scheduling problem for unit-length jobs, where each job is specified by its release time, deadline, and a nonnegative weight (QoS value). The goal is to maximize the *weighted throughput*, that is the total weight of scheduled jobs.

We first give a randomized algorithm RMIX with competitive ratio of $e/(e-1) \approx 1.582$. This is the first algorithm for this problem with competitive ratio smaller than 2. Then we consider s-bounded instances where the span of each job (deadline minus release time) is at most s. We give a 1.25-competitive randomized algorithm for 2-bounded instances, matching the known lower bound. We give a deterministic algorithm EDF_α, whose competitive ratio on s-bounded instances is

*Institute of Computer Science, The Hebrew University of Jerusalem, Israel. {yair,tron}@cs.huji.ac.il

†Department of Computer Science and Information Systems, The University of Hong Kong, Hong Kong. Supported by an RGC research grant. {chin,pyfung}@csis.hku.hk

‡Department of Computer Science, University of California, Riverside, CA 92521. Supported by NSF grants CCR-9988360 and CCR-0208856. {marek,wojtek}@cs.ucr.edu

§Mathematical Institute, AS CR, Žitná 25, CZ-11567 Praha 1, Czech Republic. Email {sgall,tichy}@math.cas.cz. Partially supported by Institute for Theoretical Computer Science, Prague (project LN00A056 of MŠMT ČR) and grant 201/01/1195 of GA ČR.

¶Partially supported by cooperative grant KONTAKT-ME476/CCR-9988360-001 from MŠMT ČR and NSF.

at most $2 - 2/s + o(1/s)$. For 3-bounded instances its ratio is $\phi \approx 1.618$, matching the lower bound. Previously, an upper bound of ϕ was known for 2-bounded instances, and our work extends this result.

Next, we consider 2-uniform instances, where the span of each job is exactly 2. We prove a lower bound of $4 - 2\sqrt{2} \approx 1.172$ for randomized algorithms. For deterministic memoryless algorithms, we prove a lower bound of $\sqrt{2} \approx 1.414$, matching a known upper bound.

Finally, we consider the multiprocessor case and give an $1/(1 - (\frac{M}{M+1})^M)$-competitive algorithm for M processors. We also show improved lower bounds for the general and 2-uniform cases.

1 Introduction

Network protocols today offer only the 'best-effort service', the term – misnomer, in fact – that describes the most basic level of service that does not involve firm guarantees for packet delivery. Next-generation networks, however, will provide support for differentiated services, to meet various quality-of-service (QoS) demands from the users. In this paper we consider an online buffer management problem that arises in such QoS applications.

In the *bounded delay buffer problem* [15, 4], packets arrive and are buffered at network switches. At each integer time step, only one packet can be sent along the link. Each packet is characterized by its QoS value, which can be thought of as a benefit gained by forwarding the packet. Network switches can use this QoS value to prioritize the packets. Each packet has a deadline that indicates the latest time when a packet can be sent. In overload conditions, some packets will not be sent by their deadline. Such packets do not contribute to the benefit value, and can as well be dropped. The objective is to maximize the total value of the forwarded packets, that is the *weighted throughput*.

It is easy to see that this buffer management problem is equivalent to the following unit-job scheduling problem. We are given a set of n unit-length jobs, with each job j specified by a triple (r_j, d_j, w_j) where r_j and d_j are integral release times and deadlines, and w_j is a non-negative real weight. One job can be processed at each integer time. We use the term *weighted throughput* or *gain* for the total weight of the jobs completed by their deadline. The goal is to compute a schedule for the given set of jobs that maximizes the weighted throughput. In Graham's notation, this problem can be described as $1|p_j = 1, r_j| \sum_j w_j U_j$.

In this paper we focus on the online version of this problem, where each job arrives at its release time. At each time step, an online algorithm needs to schedule one of the pending jobs, without the knowledge of the jobs released later in the future. An online algorithm $\mathcal{A}$ is called R-competitive, if its gain on any instance is at least $1/R$ times the optimum (offline) gain on this instance. The smallest such value R is called the *competitive ratio* of $\mathcal{A}$. The competitive ratio is commonly used as a performance measure for online algorithms, and we adopt this measure in this paper.

For unit jobs, some restrictions on instances have been proposed in the literature [15, 4, 11]. In *s-bounded instances*, the span of the jobs (defined as the difference between the deadline and the release time) is at most s, and in *s-uniform instances* the span of each job is exactly s. In the context of QoS buffer management, these cases correspond to QoS situations in which the end-to-end delay is critical and only a small amount of delay is allowed at each node [15].

The unit-job scheduling problem is related to another scheduling problem which also arises from QoS applications. In *metered-task model* [8, 12], each job is specified by four real numbers: release time, deadline, processing time (not necessarily unit), and weight. Preemptions are allowed. Unlike in classical scheduling, even non-completed jobs contribute to the overall gain. Specifically, the gain of a job is proportional to the amount of it that was processed. This problem arises as a QoS problem of transferring large images over a network of low bandwidth [8]. It is also related to a problem called *imprecise computation* in real-time systems, and has applications in numerical computation, heuristic search, and database query processing (see [16]).

Past work. A naive greedy algorithm that always schedules the heaviest job is known to be 2-competitive [15, 14]. No better algorithm, deterministic or randomized, is known for the general case. For the deterministic case, a lower bound of $\phi \approx 1.618$ was shown in [4, 11, 14]. In the randomized case, [11] give a lower bound of 1.25. (The proof in [11] was for metered tasks, but it carries over to unit jobs.) Both of those lower bounds apply even to 2-bounded instances.

For the 2-bounded case, a ϕ-competitive (and thus optimal) algorithm was presented in [15]. Deterministic algorithms for 2-uniform instances were studied by [4], who established a lower bound of $\frac{1}{2}(\sqrt{3}+1) \approx 1.366$ and an upper bound of $\sqrt{2} \approx 1.414$.

In [15], a version of the buffer management problem was studied in which the output port has *bandwidth* M (that is, M packets at a time can be sent). This corresponds to the problem of scheduling unit-time jobs on M processors. In [15] a lower bound of $4-2\sqrt{2}$ was presented that applies even to the 2-bounded model. For the 2-uniform case, a lower bound of 10/9 was given.

Other models of buffer management were considered in the literature [17, 2]. For example, in the FIFO model, the buffer has finite capacity, and packets must be forwarded in the same order as they arrive, although some packets can be dropped (in the preemptive version). An online algorithm needs to decide which packets to admit into the buffer, and then which packets to drop. In [5, 6] switches with many input queues and a single output queue have been considered.

Our results. First, we give a randomized algorithm with competitive ratio $e/(e-1) \approx 1.582$, which is the first algorithm for this problem with competitive ratio below 2. Our algorithm has been inspired by the techniques developed in [12] for metered tasks.

For 2-bounded instances, we give a 1.25-competitive randomized algorithm, matching the known lower bound from [11].

We also give a deterministic algorithm EDF_α whose competitive ratio on 3-bounded instances is $\phi = 1.618$, matching the lower bound. This result extends previous results from the literature for 2-bounded instances [15], and it provides evidence that a ϕ-competitive deterministic algorithm might be possible for the general case. For 4-bounded instances, EDF_α is $\sqrt{3} \approx 1.732$ competitive, and for s-bounded instances it is $2 - 2/s + o(1/s)$ competitive. However, without the restriction on the span, it is only 2-competitive.

For 2-uniform instances, we prove a lower bound of $4 - 2\sqrt{2} \approx 1.172$ for randomized algorithms, improving the 10/9 bound from [15]. In the deterministic case, we prove a lower bound of $\sqrt{2} \approx 1.414$ on algorithms that are memoryless and scale-invariant (that is, the algorithm's decision is invariant under multiplying all weights of pending jobs by a constant.) This matches the previously known upper bound in [4]. We remark that all competitive algorithms for unit-job scheduling in the literature are memoryless and scale-invariant.

The known bounds are summarized in the table below. Our results are marked with [$*$]. A blank entry indicates that the bound in this entry follows from another bound in the same column.

	deterministic		randomized	
	upper bound	lower bound	upper bound	lower bound
General	2 [15]		1.582... [$*$]	
s-bounded	$2 - 2/s + o(1/s)$ [$*$]			
4-bounded	1.732... [$*$]			
3-bounded	1.618... [$*$]			
2-bounded	1.618... [15]	1.618... [4, 11]	1.25 [$*$]	1.25 [11]
2-uniform	1.414... [4]	1.366... [4]		1.172.... [$*$]

Finally, we study the multiprocessor case, $M|\text{online-time}, p_j = 1, r_j| \sum_j w_j U_j$ in Graham's notation, that corresponds to the buffer management problem in which the output port has *bandwidth* M, meaning that it can send M packets at a time. We give a $1/(1 - (\frac{M}{M+1})^M)$-competitive algorithm for the case of M processors. For randomized algorithms, we also show improved lower bounds of 1.25 for the general and $4 - 2\sqrt{2} \approx 1.172$ for the 2-uniform cases.

In addition to those results, we introduce a new algorithm called Bal_β, where β is a parameter, and we analyze it in several cases. On 2-uniform instances, $\text{Bal}_{\sqrt{2}/2}$ is $\sqrt{2}$-competitive, matching the bound from [4]. On 2-bounded instances, Bal_β is ϕ-competitive (and thus optimal) for two values of $\beta \in \{2 - \phi, \phi - 1\}$. It is also ϕ-competitive for 3-bounded instances. Although we can show that Bal_β cannot be ϕ-competitive in general, we conjecture that for some values of β its ratio is better than 2.

Our results show the power of randomization for the problem of scheduling unit jobs. For the general version, our randomized algorithm outperforms all deterministic algorithms, even on the special case of span 2. For span 2, we give a tight analysis of the randomized case, showing a surprisingly low competitive ratio of 1.25, compared to 1.618 in the deterministic case.

Our understanding of the power of randomization in online problems, and scheduling in particular, is still very limited. The best known online problem where randomization provably helps and the analysis of the competitive ratio of randomized algorithms is tight is caching [13, 18, 1]. In the area of scheduling, for the basic problem of makespan scheduling on identical machines, the case of two machines is fully understood [7], there are some partial results for a small number of machines [19], and a randomized algorithm for the general case that outperforms the known deterministic algorithms was found only recently [3]. We believe that our results and techniques contribute to better understanding of randomization in the online setting.

2 Preliminaries

Unit job scheduling. As we noted in the introduction, the QoS buffer management problem is equivalent to the unit-job scheduling problem. We will henceforth use job scheduling terminology in this paper. We number the jobs $1, 2, \ldots, n$. Each job j is specified by a triple (r_j, d_j, w_j), where r_j and d_j are integral release times and deadlines, and w_j is a non-negative real weight. A *schedule* S specifies which jobs are executed, and for each executed job j it specifies an integral time t when it is scheduled, where $r_j \leq t < d_j$. Only one job can be scheduled at any given time step. The *throughput* or *gain* of a schedule S on instance I, denoted $gain_S(I)$, is the total weight of the jobs in I that are executed in S. Similarly, if $\mathcal{A}$ is a scheduling algorithm, $gain_{\mathcal{A}}(I)$ is the gain of the schedule computed by $\mathcal{A}$ on I. The optimal gain on I is denoted by $opt(I)$. We say that an instance is *s-bounded* if $d_j - r_j \leq s$ for all jobs j. Similarly, an instance is *s-uniform* if $d_j - r_j = s$ for all jobs j. The difference $d_j - r_j$ is called the *span* of a job j. A job i is *pending* in schedule S at t if $r_i \leq t < d_i$ and i has not been scheduled before t.

Earliest-deadline schedules. We often assume that (offline) schedules are *(canonical) earliest-deadline*. In such schedules, for each time t, the job that is scheduled at t is chosen (from the ones that are executed in the schedule) as the pending job with the earliest deadline. Any schedule can be easily converted into an earliest-deadline schedule by rearranging its jobs. To break ties, jobs with the same deadline are ordered in order of decreasing weights. (Jobs with equal weights are ordered arbitrarily, but consistently, namely our algorithms and the adversary are assumed to break such ties in the same way; alternatively we may assume that no weights are exactly equal.)

Competitive analysis. We often view the behavior of an online algorithm $\mathcal{A}$ as a game between $\mathcal{A}$ and an adversary. Both algorithms schedule jobs released by the adversary who tries to maximize the ratio $opt(I)/gain_{\mathcal{A}}(I)$. Most of the proofs are based on a *potential function* argument. In such proofs, we define a potential function Φ that maps all possible configurations into real numbers. At each time step, an online algorithm and the adversary execute a job. The proofs are based on the following lemma which can be proven by a simple summation over all steps.

Lemma 2.1 *Let $\mathcal{A}$ be an online algorithm for scheduling unit jobs. Let Φ be a potential function that is 0 on configurations with no pending jobs, and at each step satisfies $R \cdot \Delta gain_{\mathcal{A}} \geq \Delta adv + \Delta\Phi$, where $\Delta\Phi$ represents the change of the potential, and $\Delta gain_{\mathcal{A}}$, Δadv represent $\mathcal{A}$'s and the adversary gain in this step. Then $\mathcal{A}$ is R-competitive.*

The lemma above applies to randomized algorithms as well. In that case, however, $\Delta gain_{\mathcal{A}}$ and $\Delta\Phi$ are the *expected* values of the corresponding quantities, with respect to the algorithm's random choices at the given step.

In some proofs we use a different approach called *charging*. In a charging scheme, the weight of each of the jobs in the adversary schedule is charged to a job, or several jobs, in our schedule, in such a way that each job in our schedule is charged at most R times its weight. If such a charging scheme exists, it implies that our algorithm is R-competitive.

Metered tasks. As discussed in the introduction, our problem is related to the metered-task model. Consider the *discrete* metered-task model, in which jobs have integral release times, deadlines and processing lengths, and the algorithm can only switch jobs at integral times. (In [11] this model is called *non-timesharing*.) Then:

Theorem 2.2 *The unit-job scheduling problem with a single processor is equivalent to the single processor discrete metered-task model. The unit-job scheduling problem with M processors is a special case of the M-processor discrete metered-task model (assuming jobs are migratory); they are equivalent when, in addition, all jobs in the metered-task model are of unit length.*

The theorem can be easily proved by observing that each job in the discrete metered-task model of length p can be transformed into p unit-length jobs, each having the same release time, deadline and weight as the original job. Note that for the multiprocessor case and arbitrary jobs, the problems are not equivalent because a long job cannot be processed by several processors simultaneously, while a number of different unit-length jobs can be processed in this way.

The continuous version of the metered-task model [8, 12, 11] bears some resemblance to the randomized case of unit-job scheduling, although it is not clear whether the results from the former model can be automatically translated into results for the latter model. One may attempt to convert a deterministic algorithm $\mathcal{D}$ for metered tasks into a randomized $\mathcal{R}$ algorithm for unit jobs, by setting the probability of $\mathcal{R}$ executing a given job j to be equal to $\mathcal{D}$'s fraction of the processor power devoted to j. It is not hard to see, however, that, in general, the expected gain of $\mathcal{R}$ will not be the same as the gain of $\mathcal{D}$.

3 Randomized Algorithm RMix

In this section we give a randomized algorithm for scheduling unit jobs with competitive ratio $e/(e-1) \approx 1.582$. This is the first algorithm for unit jobs with competitive ratio smaller than 2.

Algorithm RMix**.** At each step, we inductively select a sequence of pending jobs $h_1, \ldots, h_k$ as follows: (i) h_1 is the heaviest job, (ii) h_{i+1} is the heaviest job j such that $w_j > w_{h_1}/e$ and $d_j < d_{h_i}$; if such j does not exist, we set $k = i$. In case of ties, prefer jobs with earlier deadlines. Denote $v_i = w_{h_i}$ for $i = 1, \ldots, k$ and $v_{k+1} = w_{h_1}/e$. The algorithm executes job h_i with probability $\delta_i = \ln(v_i) - \ln(v_{i+1})$. (Note that $\sum_{i=1}^{k} \delta_i = \ln v_1 - \ln v_{k+1} = 1$.)

Theorem 3.1 *Algorithm* RMix *is* $\frac{e}{e-1} \approx 1.582$*-competitive.*

Proof: At a given time step, let X be the set of pending jobs in RMix, and let Y be the set of pending jobs in the adversary schedule that the adversary will schedule in the future. We assume that the adversary schedule is canonical earliest-deadline.

Define the potential $\Phi = \sum_{i \in Y - X} w_i$. Job arrivals and expirations cannot increase the potential as these jobs are not in $Y - X$: the arriving job is always in X and the expiring job is never in Y by the definition of Y. So we only need to analyze how the potential changes after job execution. Consider a given time step. Using the notation from the algorithm, the expected gain of RMix is $\omega = \sum_{i=1}^{k} \delta_i v_i$. Thus, by Lemma 2.1, denoting by j the job scheduled by the adversary, it sufficient to prove that $w_j + \Delta\Phi \le \frac{e}{e-1}\omega$.

Suppose first that $j \in Y - X$. Executing j by the adversary decreases Φ by w_j. At the same time, one of $h_1, \ldots, h_k$ may be added to $Y - X$, h_i with probability δ_i, increasing the expected value of Φ by at most ω. So $\Delta\Phi \le -w_j + \omega$. It follows that $w_j + \Delta\Phi \le \omega \le \frac{e}{e-1}\omega$.

Now assume that $j \in Y \cap X$. Inequality $\ln x \le x - 1$ for $x = v_{i+1}/v_i$ implies that, for any $i \le k$,

$$v_i - v_{i+1} \;\le\; v_i(\ln v_i - \ln v_{i+1}) \;=\; \delta_i v_i. \tag{1}$$

We have $w_j \leq v_1$ as $j \in X$. Let $p \in \{1, \ldots, k+1\}$ be the largest index such that $w_j \leq v_p$. By the assumption that the adversary schedule is earliest-deadline, we know that he will not execute any h_i, $i = p, \ldots, k$, in the future, so these are not in Y. The expected increase of Φ is then at most $\sum_{i=1}^{p-1} \delta_i v_i$. So, using $v_{k+1} = v_1/e$ and (1), we have $w_j + \Delta\Phi \leq v_p + \sum_{i=1}^{p-1} \delta_i v_i = \frac{e}{e-1}(v_p - v_{k+1}) + \frac{1}{e-1}(v_1 - v_p) + \sum_{i=1}^{p-1} \delta_i v_i = \frac{e}{e-1} \sum_{i=p}^{k}(v_i - v_{i+1}) + \frac{1}{e-1} \sum_{i=1}^{p-1}(v_i - v_{i+1}) + \sum_{i=1}^{p-1} \delta_i v_i \leq \frac{e}{e-1} \sum_{i=p}^{k} \delta_i v_i + \frac{1}{e-1} \sum_{i=1}^{p-1} \delta_i v_i + \sum_{i=1}^{p-1} \delta_i v_i = \frac{e}{e-1}\omega$. □

4 An Optimal Randomized Algorithm for 2-Bounded Instances

In this section we give a randomized algorithm for 2-bounded instances with competitive ratio 1.25. This matches the lower bound from [11], and thus completely resolves this case. To simplify terminology and notation, we will use the weights of jobs to identify jobs. Thus, we will say "job w" meaning "the job with weight w".

Algorithm R2B. Define $p_{ab} = 1$ if $a \geq b$ and $p_{ab} = \frac{4a}{5b}$ otherwise. Let $q_{ab} = 1 - p_{ab}$. Let a and b denote the heaviest jobs of span 1 and span 2, respectively, released at this time step. If the currently pending job is x, let $u = \max(x, a)$. Execute u with probability p_{ub} and b with probability q_{ub}.

Theorem 4.1 *Algorithm* R2B *is* 1.25*-competitive.*

Proof: Without loss of generality, we can assume that at each step (except last) exactly one job of span 1 is issued. All jobs of span 1 except the heaviest one can be simply ignored; if no job is issued, we treat it as a job of weight 0. Similarly, we can assume that at each step (except last) exactly one job of span 2 is issued. This can be justified as follows: If, at a given time t, the optimum schedule contains a job of span two released at t, we can assume that it is the heaviest such job. A similar statement holds for Algorithm R2B, since its decision at each step depends only on the heaviest job of span 2. Thus all the other jobs of span 2 can be ignored in this step, and treated as if they are issued with span 1 in the following time step.

First note that p_{ab} satisfies the following properties for any $a, b \geq 0$.

$$5p_{ab}a \geq 4a - b \qquad (2) \qquad\qquad 5p_{ab}a + 2q_{ab}b \geq 4a \qquad (4)$$
$$5(p_{ab}a + q_{ab}b) \geq 4b \qquad (3) \qquad\qquad 5p_{ab}a + 2q_{ab}b \geq b \qquad (5)$$

Algorithm R2B is memoryless and randomized, so its state at each step is given by a pair $\langle x, s\rangle$, where x is the job of span 2 issued in the previous step, and s is the probability that x was executed in the previous step (i.e., no job is pending). Denote $t = 1 - s$ the probability that x is pending.

Denoting by $z \in \{0, x\}$ the pending job of the adversary, the complete configuration at this step is described by a triple $\langle x, s, z\rangle$. Let Φ_{xsz} denote the potential function in the configuration $\langle x, s, z\rangle$. We put $\Phi_{xs0} = 0$ and $\Phi_{xsx} = \frac{1}{4}x \cdot \max(5s - 1, 3s)$.

Consider one step, where the configuration is $\langle x, s, z\rangle$, two jobs a, b are issued, of span 1 and span 2, respectively. The new configuration is $\langle b, s', z'\rangle$, where $s' = sq_{ab} + tq_{x'b}$, $x' = \max(a, x)$, and $z' \in \{0, b\}$. Using Lemma 2.1, we need to show that for each adversary move:

$$R \cdot \Delta gain_{\text{R2B}} - \Phi_{bs'z'} + \Phi_{xsz} \quad \geq \quad \Delta adv \tag{6}$$

where $\Delta gain_{\text{R2B}}$ is the expected weight of a job scheduled by R2B and Δadv the weight of the job scheduled by the adversary.

<u>Case 1</u>: Adversary schedules b. Then $\Phi_{xsz} \geq 0$, $\Delta adv = b$, $z' = 0$, and $\Phi_{bs'z'} = 0$. For a fixed value of u in the algorithm, the expected gain of the algorithm is $p_{ub}u + q_{ub}b$ and (3) implies $\frac{5}{4}(p_{ub}u + q_{ub}b) \geq b$. By averaging over $u \in \{a, x'\}$ we get $R \cdot \Delta gain_{\text{R2B}} \geq b$, which implies (6).

<u>Case 2</u>: Adversary does not schedule b. Then $z' = b$, $\Phi_{bs'z'} = \frac{1}{4}b \cdot \max(5s' - 1, 3s')$, and $\Delta gain_{\text{R2B}} = s'b + sp_{ab}a + tp_{x'b}x'$. Substituting into (6), it is sufficient to prove that

$$\min(b, 2s'b) + 5sp_{ab}a + 5tp_{x'b}x' + 4 \cdot \Phi_{xsz} \geq 4 \cdot \Delta adv \tag{7}$$

<u>Case 2.1</u>: Adversary schedules a. Then $\Delta adv = a \leq x'$ and $\Phi_{xsz} \geq 0$. For the first term of the minimum, we use (2) twice and get $b + 5sp_{ab}a + 5tp_{x'b}x' = s(b + 5p_{ab}a) + t(b + 5p_{x'b}x') \geq 4sa + 4tx' \geq 4a$. For the second term of the minimum, we use (4) twice and get $2s'b + 5sp_{ab}a + 5tp_{x'b}x' = s(5p_{ab}a + 2q_{ab}b) + t(5p_{x'b}x' + 2q_{x'b}b) \geq 4sa + 4tx' \geq 4a$

<u>Case 2.2</u>: Adversary schedules $z = x$. It must be the case that $x' = x \geq a$, as otherwise the adversary would prefer to schedule a. We have $\Delta adv = x$.

If $x \geq b$, then $p_{xb} = 1$. We use $4\Phi_{xsz} = 4\Phi_{xsx} \geq (5s - 1)x$ and obtain $5tp_{xb}x + 4\Phi_{xsz} \geq 5tx + 5sx - x = 4x$, which implies (7).

It remains to consider the case $x < b$. Using (2), (5) and (4) we obtain $b + 5tp_{xb}x \geq b + t(4x - b) = 4tx + sb$ and $2s'b + 5sp_{ab}a + 5tp_{xb}x = s(2q_{ab}b +$

$5p_{ab}a) + t(2q_{xb}b + 5p_{xb}x) \geq sb + 4tx$. Together with with $4\Phi_{xsz} = 4\Phi_{xsx} \geq 3sx$ and $x < b$ this implies $\min(b, 2s'b) + 5sp_{ab}a + 5tp_{xb}x + 4\Psi_{xsz} \geq 4tx + sb + 3sx \geq 4x$ and (7) follows. □

5 Deterministic Algorithm for s-Bounded Instances

The 2-bounded (deterministic) case is now well understood. A ϕ-competitive algorithm was given in [4], matching the lower bound from [15, 14]. In this section, we extend the upper bound of ϕ to 3-bounded instances.

For the general case, the best known competitive ratio for deterministic algorithm is 2, attained for example by a simple greedy algorithm which always schedules the heaviest job [15, 14]. It is known that this algorithm, as well as other natural ones, is not better than 2-competitive.

We define two algorithms. They both use a real-valued parameter denoted α and β and they are both ϕ-competitive for 3-bounded instances for an appropriate value of the parameter. In this section, h always denotes the heaviest pending job.

The first algorihtm simply schedules a relatively heavy job with the smallest deadline:

Algorithm EDF_α**:** Let f be the earliest-deadline job such that $w_f \geq \alpha w_h$. Execute f.

We prove that $\text{EDF}_{\phi-1}$ is ϕ-competitive for 3-bounded instances.

The idea of the second algorithm is to balance the maximum gain in the next step against the discounted projected gain in the following steps, if no new jobs are issued. A *plan* (at time t) is an optimal schedule of jobs pending at time t. A plan can be computed by iteratively scheduling pending jobs, from heaviest to lightest, at their latest available slots.

Algorithm BAL_β**.** At each step, execute the job j that maximizes $w_j + \beta\pi_j$, where π_j is the total weight of the plan in the next time step, if j is executed in the current step. (In a case of a tie, the algorithm chooses the earliest-deadline job, and if there are several, the heaviest one among those.)

We conjecture that BAL is better than 2-competitive for general instances. We establish the following facts about BAL_β. All positive results can be shown using Lemma 2.1. In the following x is a pending job of BAL_β and y is a pending job of the adversary.

- Let $\beta \in \{\phi - 1, 2 - \phi\}$, where $\phi \approx 1.618$ is the golden ratio. Then $\textsc{Bal}_\beta$ is ϕ-competitive for 2-bounded instances. For $\beta = 2 - \phi = 1 - 1/\phi$, the function $\Phi = [y - x]^+$ can be used as potential, where $[z]^+ = \max(z, 0)$; for $\beta = \phi - 1$, use $\Phi = \max(x, y) - \phi x$.
- $\textsc{Bal}_{\sqrt{2}/2}$ is $\sqrt{2}-$competitive for 2-uniform instances (this is the best ratio for memoryless algorithms, as discussed in Section 6). The proof uses the potential $\Phi = (\sqrt{2} - 1)[y - 2x + \sqrt{2}(y - x)^+]$.
- $\textsc{Bal}_{\phi-1}$ is $\phi-$competitive for 3-bounded instances. The proof is quite complicated and omitted.
- $\textsc{Bal}_{\phi-1}$ is not $\phi-$competitive for 8-bounded instances. See Appendix B.

Theorem 5.1 $\textsc{Edf}_{\phi-1}$ *is ϕ-competitive for 3-bounded instances.*

Proof: We fix a canonical earliest-deadline adversary schedule A. Let E be the schedule computed by $\textsc{Edf}_{\phi-1}$. We use the following charging scheme: Suppose that j is the job scheduled by the adversary at time t. If j is executed in E before time t, charge j to its copy in E. Otherwise, charge j to the job in E scheduled at time t.

Fix some time step t, and let f be the job scheduled in E at time t. Let also h be the heaviest pending job in E at time t. By the definition of $\textsc{Edf}_{\phi-1}$, f is the earliest-deadline job with $w_f \geq (\phi - 1)w_h = w_h/\phi$. Denote also by j the job scheduled in A at time t.

Job f receives at most two charges: one from j and one from itself, if f is executed in A at some later time. Ideally, we would like to prove that the sum of the charges is at most ϕw_f. It turns out that in some cases this is not true, and, if so, we then show that for the job g scheduled by E in the next step, the total of all charges to f and g is at most $\phi(w_f + w_g)$. Summing over all such groups of one or two jobs, the ϕ-competitiveness of $\textsc{Edf}_{\phi-1}$ follows.

If f receives only one charge, it is at most ϕw_f: If this charge is from f, it is trivially at most w_f. If the charge is from j (not scheduled before t in E), then j is pending at t in E and thus $w_j \leq w_h \leq \phi w_f$, by the definition of $\textsc{Edf}_{\phi-1}$. In this case the group consist of a single job and we are done.

It remains to handle the case when f receives both charges. Since in A job j is before f, we have $d_j \leq d_f$ (and for $d_j = d_f$, the tie is broken in favor of j – recall our tie-breaking convention.) But at time t, $\textsc{Edf}_{\phi-1}$ chooses f, so j is not eligible for execution by $\textsc{Edf}_{\phi-1}$, that is $w_j < (\phi - 1)w_h$.

If $w_f = w_h$, then f is charged at most $w_f + w_j \leq (1 + \phi - 1)w_h = \phi w_f$, and we have a group with a single job again.

Otherwise, $w_f < w_h$ and the adversary does not schedule f at time t, hence $d_f \geq t+2$. By the rule of $\mathrm{EDF}_{\phi-1}$, $d_h > d_f$. As the span is bounded by 3, it has to be the case that $d_h = t+3$ and $d_f = t+2$. Thus the adversary schedules f at time $t+1$. The weight of the job g scheduled at time $t+1$ in E is $w_g \geq (\phi-1)w_h$, as $h \neq f$ is still pending in E. Furthermore, g gets only the charge from itself, as the adversary at time $t+1$ schedules f which is charged to itself. The total weight of the jobs charged to f and g is thus at most $w_j + w_f + w_g \leq (\phi-1)w_h + w_f + w_g \leq \frac{3}{2}(w_f + w_g)$, since both w_h and w_f are at least $(\phi-1)w_h$. In this case we have a group of two jobs. □

Extending the idea in the proof above easily yields an upper bound of $2-\Theta(1/s)$ on the competitive ratio of EDF_α on s-bounded instances, for an appropriately chosen α. The proof of the following more precise bound is presented in Appendix A.

Theorem 5.2 *For each $s \geq 4$, algorithm $\mathrm{EDF}_{1/\lambda_s}$ is λ_s-competitive for s-bounded instances, where λ_s is the unique non-negative solution of equation*

$$(2-\lambda_s)(\lambda_s^2 + \lfloor \frac{s}{3} \rfloor \lambda_s + s - 2 - 2\lfloor \frac{s}{3} \rfloor) \;\; = \;\; \lambda_s^2 - \lambda_s.$$

We get $\lambda_4 = \sqrt{3} \approx 1.732$. For larger s, the equation is cubic. It can be verified that $2 - \frac{2}{s} \leq \lambda_s \leq 2 - \frac{1}{s}$, and in the limit for $s \to \infty$, $\lambda_s = 2 - 2/s + o(1/s)$. Some numerical values are given in Table 1.

Recall that, by Theorem 2.2, results for discrete metered tasks can be applied to unit-job scheduling. Here we describe two such results. We say a pending job i *dominates* another pending job j if $d_i \leq d_j$ and $w_i > w_j$. A pending job is *dominant* if no other pending job dominates it. In [10], the authors considered the case of the metered-task model when there are at most s dominant jobs at each time, and proposed an online algorithm GAP for this case. In s-bounded instances there can be at most s pending dominant jobs at any time, since there can be at most one dominant job for each span value. Thus, the results from [10] imply that:

Theorem 5.3 GAP *is r_s-competitive for s-bounded instances, where r_s is the unique positive real root of the equation $r_s = 1 + r_s^{-1/(s-1)}$.*

We can show that $r_s = 2 - \Theta(\frac{1}{s})$. Table 1 gives a comparison of EDF_α (with $\alpha = 1/\lambda_s$) and GAP. EDF_α has a smaller competitive ratio for s-bounded instances, on the other hand GAP can be applied also to the more general set of instances that have at most s dominant jobs at any time. GAP can also be slightly modified to give the same performance without knowing the value of s in advance. In [9], an algorithm FIT was presented for the

s	2	3	4	5	6	7	10	20	∞
EDF$_\alpha$	1.618	1.618	1.732	1.769	1.791	1.813	1.856	1.917	2
GAP	1.618	1.755	1.819	1.857	1.881	1.899	1.930	1.965	2

Table 1: Comparison of the upper bounds for EDF and GAP.

discrete metered-task model. Its competitive ratio is better than 2 when the *importance ratio*, that is the ratio of maximum to minimum job weights, is at most ξ (without any restriction on s). By Theorem 2.2, we have:

Theorem 5.4 FIT *is* $(2-1/(\lceil \lg \xi \rceil + 2))$*-competitive for unit-job scheduling.*

6 2-Uniform Instances

In this section we consider 2-uniform instances, where $d_j = r_j + 2$ for each job j. We first prove a lower bound of $4 - 2\sqrt{2} \approx 1.172$ on the competitive ratio of randomized algorithms. This improves a lower bound of 10/9 from [15].

Theorem 6.1 *No randomized algorithm can be better than* $(4 - 2\sqrt{2})$*-competitive for 2-uniform instances.*

Proof: We use Yao's minimax principle [20], by showing a distribution on instances that forces each online algorithm $\mathcal{A}$ to have ratio at least $4 - 2\sqrt{2}$.

We will generate an instance randomly. Fix a large integer n and let $a = \sqrt{2} + 1$ and $p = 1/a = \sqrt{2} - 1$. Each instance consists of stages $0, 1, \ldots,$ where in stage j we have three jobs: two jobs j, j' of weight a^j issued at time $2j$ and one job j'' of weight a^{j+1} issued at time $2j+1$. After each stage $j \le n$, we continue with probability p or stop with probability $1-p$. After stage n, at time $2n+2$, we issue two jobs of weight a^{n+1}, and stop.

Fix a deterministic online algorithm $\mathcal{A}$. We compute the expected gain of $\mathcal{A}$ and the adversary in stage $j \le n$, conditioned on stage j being reached. At time $2j$, $\mathcal{A}$ executes a job of weight a^j (it has no choice), say j. If it executes j'' at time $2j+1$, its gain in stage j is $a^j + a^{j+1} = (1+a)a^j = (2+\sqrt{2})a^j$. If it executes j', its gain is either $2a^j + a^{j+1}$ or $2a^j$, depending on whether we stop, or continue generating more stages. Thus its expected gain is $(1-p)\cdot(2a^j + a^{j+1}) + p \cdot 2a^j = (2+\sqrt{2})a^j$, same as in the previous case. Since the probability of reaching this stage is p^j, the contribution of this stage to $\mathcal{A}$'s expected gain is $2+\sqrt{2}$.

We now calculate the adversary gain in stage j (again, condition on this stage being generated). If we stop, the adversary gains $2a^j + a^{j+1}$, otherwise he gains $a^j + a^{j+1}$, so his expected gain is $(1-p)\cdot(2a^j+a^{j+1})+p\cdot(a^j+a^{j+1}) = a^j(2-p+a) = 4a^j$. Thus the contribution of this stage to the adversary's gain is 4.

Summarizing, for each step, except the last one, the contributions towards the expected value are $2+\sqrt{2}$ for $\mathcal{A}$ and 4 for the adversary independent of j. The contributions of stage $n+1$ are different, but also constant. So the overall ratio will be, in the limit for $n \to \infty$, the same as the ratio of the contributions of stages $0, \ldots, n$, which is $4/(2+\sqrt{2}) = 4 - 2\sqrt{2}$, as claimed. $\square$

Deterministic algorithms for 2-uniform instances were studied in [4], where an upper bound of $\sqrt{2}$ was given. The same upper bound can be achieved using algorithm $\text{BAL}_{\sqrt{2}/2}$. As we show below, it is not possible to beat ratio $\sqrt{2}$ by choosing a different parameter β in BAL_β, nor with any deterministic memoryless, scale-invariant algorithm. Recall that we define an online algorithm $\mathcal{A}$ to be *memoryless* if its decision at each step depends only on the pending jobs (including those released at the current step.) We say that $\mathcal{A}$ is scale-invariant if its decision is invariant under scaling of weights (multiplying the weights of all pending jobs by a positive real number does not affect the choice of the job to be executed).

Theorem 6.2 *No deterministic memoryless algorithm can achieve competitive ratio better than $\sqrt{2}$ for 2-uniform instances.*

Proof: Let $\mathcal{A}$ be a deterministic memoryless algorithm. Adversary will issue at most one pending job at each step. Suppose that $y > 0$ is a pending job (or, more precisely, the weight of the pending job). We will have two types of moves:

(a) Issue two jobs with weight y.

(b) Issue one job with weight $b > y$.

Note that in step (a), $\mathcal{A}$ has no choice but to execute a job of weight y. Further, it does not matter which job it chooses for execution. Consider steps of type (b). $\mathcal{A}$'s decision is only a function of b/y. For any $\epsilon > 0$ there are two constants a_1, a_2 with $|a_1 - a_2| \leq \epsilon$, such that $\mathcal{A}$ executes y when $b/y = a_1$ and b when $b/y = a_2$. (If one of a_1, a_2 does not exist, $\mathcal{A}$ is not competitive at all.) To simplify the proof, we will further assume that $a_1 = a_2 = a$, and allow the adversary to break the tie and determine which job will be executed by $\mathcal{A}$ when $b/y = a$. The argument below can be easily modified by replacing a by a_1 or a_2 to force $\mathcal{A}$ to make the desired decision, and then taking the limit with $\epsilon \to 0$.

Choose some large integer n. The adversary uses three instances. In instance I_1 we issue two jobs of weight 1 at step 0 and one job of weight a at step 1. The adversary forces $\mathcal{A}$ to execute job a at step 1. The gain of $\mathcal{A}$ is $1 + a$, the adversary gains $2 + a$, so the ratio is $R_1(a) = (a + 2)/(a + 1)$.

In instance I_2, we start with two jobs of weight 1 at step 0. Then, at each time $j = 1, \ldots, n$ we issue a job j of weight a^j, and at time $n+1$ we issue two jobs n', n'' of weight a^n. The adversary breaks the tie so that $\mathcal{A}$ executes job j at time $j + 1$, for $j = 1, ..., n$. So the gain of $\mathcal{A}$ is $2 + \sum_{j=1}^n a^j + a^n$, and the gain of the adversary is $1 + \sum_{j=1}^n a^j + 2a^n$. Taking the limit with $n \to \infty$, so that we can ignore low-order terms, the ratio is $R_2(a) = (3a - 2)/(2a - 1)$.

The last instance I_3 has the same jobs as I_2, except that at time $n+2$ we also issue job n^* of weight a^{n+1}. The adversary forces $\mathcal{A}$ to execute this job at time $n+2$. The gain of $\mathcal{A}$ is $2+\sum_{j=1}^n a^j + a^{n+1}$, and the gain of the adversary is $1 + \sum_{j=1}^n a^j + 2a^n + a^{n+1}$. In the limit, the ratio is $R_3(\alpha) = 1 + 2/a - 2/a^2$.

The overall ratio is $R(a) = \max\{R_1(a), R_2(a), R_3(a)\}$. By routine calculations, this ratio is minimized for $a \in \left\{\sqrt{2}, 2 + \sqrt{2}\right\}$, and for those values $R(a) = \sqrt{2}$. □

7 The Multiprocessor Case

In this section we consider the multiprocessor case. The greedy 2-competitive algorithm [15, 14] applies to both uniprocessor and multiprocessor cases. We show an algorithm that for M processors achieves competitive ratio $(1 - (\frac{M}{M+1})^M)^{-1}$, showing that with a larger number of processors we can improve the competitive ratios. When $M \to \infty$ this ratio tends to $e/(e-1) \approx 1.58$, beating the $\phi \approx 1.618$ bound for $M = 1$ [11]. The basic idea of our algorithm is similar to algorithm MIXED for metered tasks in [11]. It also has some similarity to our randomized algorithm RMIX. We divide the processing effort between M processors, such that each processor works on the earliest-deadline job with weight above a certain threshold. This threshold decreases geometrically for each processor. If no job is above the threshold, we select the heaviest remaining job, and reset the threshold to the weight of this job. Throughout this section let $\beta = M/(M+1)$, $R = (1 - (\frac{M}{M+1})^M)^{-1}$.

Algorithm DMix-M**.** Let X be the set of pending jobs at a given time t. The algorithm chooses jobs $h_1, \dots, h_M$ as shown in the pseudocode below and schedules them for execution.

```
i ← 1 : repeat
    g ← heaviest job in X − {h_1, ..., h_{i−1}} : h_i ← g : j ← i
    repeat
        i ← i + 1
        f ← earliest-deadline job in X − {h_1, ..., h_{i−1}} with w_f ≥ β^{i−j} w_g
        if f exists then h_i ← f
    until f does not exist
```

Fix a time step t. Denote $v_i = w_{h_i}$ for all i. Normalize the weights so that $v_1 = 1$. We call those h_i selected in the outer repeat loop g-jobs. We only prove the case of two g-jobs, and leave the complete proof to the full paper. Suppose the g-jobs are h_1 and h_k, $1 < k \le M$. By the choices of DMix-M, we have $v_i \ge \beta^{i-1}$ for $i \in \{1, 2, ..., k-1\}$, $v_k < \beta^{k-1}$, and $v_i \ge v_k \beta^{i-k}$ for $i \in \{k+1, ..., M\}$. We need the following lemma, with proof omitted.

Lemma 7.1 *(i)* $(k-1) + (M-k+1)v_k \le R \cdot \left(\sum_{i=0}^{k-2} \beta^i + v_k \sum_{i=k}^{M} \beta^{i-k}\right)$.
(ii) $M\beta^{p-k} v_k + \sum_{i=1}^{p} v_i \le R \cdot \sum_{i=1}^{M} v_i$ *for any positive integer* $p \in \{k, k+1, ..., M\}$.

Theorem 7.2 DMix-M *is* R*-competitive for* M *processors, where* $R = (1 - (\frac{M}{M+1})^M)^{-1}$.

Proof: (Sketch.) For a given input instance, let D be a schedule of DMix-M and A the adversary schedule. As usual, we assume that A is canonical earliest-deadline. Fix a time step t. Let

H = $\{h_1, ..., h_M\}$, the jobs executed by DMix-M at time t
J = the set of M jobs executed by the adversary at time t
X = the pending jobs of DMix-M at time t
Y = the pending jobs of the adversary at time t that will be executed at time t or later

For a set of jobs I, let $w(I) = \sum_{i \in I} w_i$ denote the total weight of I. Define the potential function $\Phi = w(Y - X)$. By Lemma 2.1, it is thus sufficient to show that $w(J) + \Delta\Phi \leq R \cdot w(H)$.

Job arrivals and expirations cannot increase the potential. So we only need to analyze how the potential changes after job executions. The change in the potential due to the adversary executing the jobs in J is $-w(J-X)$, as the jobs in $J - X$ contribute to the current potential but will not contribute in the next step. A job h_i executed by DMix-M does not contribute to the potential in the current step, but if $h_i \in Y - J$, then, in the next step, h_i will be pending in A but not in D, so it will contribute to the new potential. Thus the change due to DMix-M executing the jobs in H is $w(H \cap Y - J)$. We conclude that $\Delta\Phi = -w(J-X) + w(H \cap Y - J)$. Therefore, in order to prove the theorem it is sufficient to show that $w(J \cap X) + w(H \cap Y - J) \leq R \cdot w(H)$.

Case 1: $H \cap Y - J = \emptyset$. Jobs $j \in J \cap X$ must have weight at most 1, at most $k - 1$ of them can have weights larger than v_k, since otherwise DMix-M would choose the g-jobs differently. Thus, using Lemma 7.1, we get: $w(J \cap X) + w(H \cap Y - J) \leq (k-1) + (M - k + 1)v_k + 0 \leq R \cdot \left(\sum_{i=0}^{k-2} \beta^i + v_k \sum_{i=k}^{M} \beta^{i-k}\right) \leq R \cdot \left(\sum_{i=1}^{k-1} v_i + \sum_{i=k}^{M} v_i\right) = R \cdot w(H)$.

Case 2: $H \cap Y - J \neq \emptyset$. Let p be the largest index for which $h_p \in Y - J$. In other words, h_p is the highest-indexed job in H that will be executed in A at a later time. Since A is earliest-deadline, we have $d_j \leq d_{h_p}$ for all $j \in J$. We distinguish two subcases.

Case 2.1: $p \geq k$. In this case, $w_j < \beta^{p-k} v_k$ for any $j \in J \cap X - H$, since otherwise they would be scheduled instead of h_k. Thus by Lemma 7.1, $w(J \cap X) + w(H \cap Y - J) = w(J \cap X - H) + w(H \cap Y) \leq M\beta^{p-k} v_k + \sum_{i=1}^{p} v_i \leq R \cdot w(H)$.

Case 2.2: $p \leq k$. We have $w_j \leq v_k$ for any $j \in J \cap X - \{h_1, ..., h_k\}$, since otherwise they would be scheduled instead of h_k. Thus by Lemma 7.1 with $p = k$, $w(J \cap X) + w(H \cap Y - J) = w(J \cap X - H) + w(H \cap Y) \leq \sum_{j \in \cap X - \{h_1, \ldots, h_k\}} w_j + \sum_{i=1}^{p} v_i \leq M v_k + \sum_{i=1}^{k} v_i \leq R \cdot w(H)$. □

The lower bound proofs in [11] and Theorem 6.1 can easily be generalized to the multiprocessor case, improving the bounds in [15] ($4 - 2\sqrt{2}$ for the general case and 10/9 for the 2-uniform case):

Theorem 7.3 *No deterministic or randomized algorithm can be better than 5/4-competitive, for any number of processors M. No deterministic or ran-*

domized algorithm can be better than $4 - 2\sqrt{2}$-competitive, for 2-uniform instances on any number of processors M.

Final Remarks

We have completely resolved the 2-bounded randomized and 3-bounded deterministic cases of our problem. For general instances, we have given a randomized 1.58-competitive algorithm.

The main unit-job scheduling question, of whether there exists a $(2 - \epsilon)$-competitive deterministic algorithm, remains open, and it appears to be a challenging problem. The problem of how to tighten the gap between the bounds 1.25 and 1.58 in the randomized model also remains open.

References

[1] D. Achlioptas, M. Chrobak, and J. Noga. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234:203–218, 2000.

[2] W. A. Aiello, Y. Mansour, S. Rajagopolan, and A. Rosen. Competitive queue policies for differentiated services. In *Proc. of the IEEE INFOCOM*, pages 431–440, 2000.

[3] S. Albers. On randomized online scheduling. In *Proc. of the 34nd STOC*, pages 134–143, 2002.

[4] N. Andelman, Y. Mansour, and A. Zhu. Competitive queueing policies for QoS switches. In *Proc. of the 14th ACM-SIAM SODA*, pages 761–770, 2003.

[5] Y. Azar and Y. Richter. Management of mulit-queue switches in QoS networks. In *Proc. of the 35nd STOC*, pages 82–89, 2003.

[6] A. Bar-Noy, A. Freund, S. Landa, and J.Naor. Competitive on-line switching policies. In *Proc. of the 13th ACM-SIAM SODA*, pages 525–534, 2002.

[7] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. *Journal Computer Systems Science*, 51:359–366, 1995.

[8] E.-C. Chang and C. Yap. Competitive online scheduling with level of service. In *Proc. 7th Annual International Computing and Combinatorics Conference*, pages 453–462. Springer, 2001.

[9] F. Y. L. Chin and S. P. Y. Fung. Online scheduling with partial job values and bounded importance ratio. In *Proc. of International Computer Symposium*, pages 787–794, 2002.

[10] F. Y. L. Chin and S. P. Y. Fung. Improved competitive algorithms for online scheduling with partial job values. *9th International Computing and Combinatorics Conference, to appear*, 2003.

[11] F. Y. L. Chin and S. P. Y. Fung. Online scheduling for partial job values: does timesharing or randomization help? *Algorithmica, to appear*, 2003.

[12] M. Chrobak, L. Epstein, J. Noga, J. Sgall, R. van Stee, T. Tichy, and N. Vakhania. Preemptive scheduling in overloaded systems. In *Proc. of 29th ICALP*, pages 800–811, 2002.

[13] A. Fiat, R. Karp, M. Luby, L. A. McGeoch, D. Sleator, and N. E. Young. Competitive paging algorithms. *Journal of Algorithms*, 12:685–699, 1991.

[14] B. Hajek. On the competitiveness of online scheduling of unit-length packets with hard deadlines in slotted time. In *Conference in Information Sciences and Systems*, 2001.

[15] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. In *Proc. of the 33rd STOC*, pages 520–529, 2001.

[16] J. W. S. Liu, K.-J. Lin, W.-K. Shih, A. C. shi Yu, J.-Y. Chung, and W. Zhao. Algorithms for scheduling imprecise computations. *IEEE Computer*, 24(5):58–68, 1991.

[17] Z. Lotker and B. Patt-Shamir. Nearly optimal FIFO buffer management for DiffServ. In *In Proc. of the 21st ACM Symp. on Principles of Distributed Computing*, pages 134–143, 2002.

[18] L. McGeoch and D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6(6):816–825, 1991.

[19] S. Seiden. Online randomized multiprocessor scheduling. *Algorithmica*, 28(2):173–216, 2000.

[20] A. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. of the 18th FOCS*, pages 222–227, 1977.

A Appendix: Proof of Theorem 5.2

Proof: Throughout the proof, we put $\alpha = 1/\lambda_s$ and write and EDF instead of EDF$(1/\lambda_s)$. By EDF we also denote any schedule generated by EDF. For any time t, let M_t be the maximal weight of a job available to EDF at time t; define $M_t = 0$ if no job is available.

First we show that we can restrict ourselves to instances where all the jobs have weights α^i for some integer i. To achieve this, we slightly relax the rule of EDF. If a job with weight equal to M_t/α is available, EDF may arbitrarily choose to consider it as a candidate to be scheduled or not. Precisely, EDF schedules the earliest deadline job among those with weight strictly bigger than M_t/α (preferring heavier jobs if deadlines are equal), or an arbitrary job with earlier deadline and weight equal to M_t/α.

Call a weight bad if it is not equal to α^i for an integral i. Now we show that any instance with some bad weights can be converted to an instance with a smaller number of bad weights and with the same or larger competitive ratio on some valid run of EDF. Express each bad weight of job j as $a_j\alpha^{i_j}$ for i_j integer and $0 < a_j < \alpha$. Let $x_{min} = \min_j a_j$ and $x_{max} = \min_j(\alpha - a_j)$ (the minima are taken over all bad jobs). Now replace each weight of a bad job by $(a_j + x)\alpha^{i_j}$, for some $x \in [-x_{min}, x_{max}]$. From the definition of x_{min} and x_{max}, it follows that the order of the weights of jobs does not change, as well as the result of comparisons of one weight to another weight divided by α, except possibly for creating new ties. Consequently, any valid run of EDF on the original instance is also a valid run on the new instance, and the set of jobs scheduled in the original optimal solution gives also an optimal solution of the new instance. As the total weight of jobs scheduled both in EDF and OPT is linear in x, their ratio is monotone in $[-x_{min}, x_{max}]$ and thus it is maximized either for $x = -x_{min}$ or $x = x_{max}$. Choose the appropriate x of these two possibilities and the corresponding modified instance. By the definition of x_{min} and x_{max}, the number of bad weights has decreased.

On the restricted instances, EDF at time t schedules either a job of weight M_t or M_t/α. If a job of weight M_t/α is scheduled, the job with weight M_t remains available at time $t+1$ (if its deadline were $t+1$, EDF would schedule such a job at time t), thus $M_{t+1} \geq M_t$.

We fix an earliest-deadline schedule ADV and define a similar charging scheme as in the proof for $s = 3$. For any integer time t, suppose q is the job scheduled at t in ADV and z the job scheduled in EDF. If q is completed in EDF before time t and $w_q \geq M_t$, charge q to q in EDF. Otherwise, charge q

to z in EDF.

Each job z in EDF scheduled at time t receives charges from at most two jobs in ADV: one from z itself at t or a later time, and another one from the ADV job scheduled at t. Ideally, we would like to prove that the sum of the charges is at most αw_z. In some cases this fails, we then find a contiguous segment of EDF starting with z such that the total of all charges to the jobs in the segment is at most α times their total weight. Summing over all such groups, α competitiveness follows.

We start by two general observations.

(I) If job z scheduled at time t in EDF receives only one charge, it is at most α times its weight. If this charge is from z, it is trivially at most w_z. If the charge is from a job q scheduled before t in EDF, the charge is at most $M_t/\alpha \leq w_z$ by the definition of the charging scheme. If the charge is from a job q not scheduled before t in EDF, then q is available at t to EDF and thus $w_q \leq M_t \leq \alpha w_z$, by the definition of EDF.

(II) If job z scheduled at time t receives both charges, the charge from the job q scheduled in ADV at time t is at most M_t/α. It could be more only if q is not scheduled before t in ADV and $w_q > M_t/\alpha$; in that case q is available to EDF and has a sufficient weight, since ADV has also available both q and z and prefers q, and the ties are broken consistently, EDF also prefers q and cannot schedule z. This also implies that if $w_z = M_t$, then the z is charged at most $(1+1/\alpha)w_z \leq \alpha w_z$, using $\alpha \geq \phi$.

By the observations above, it remains to handle the case when z receives two charges and $w_z = M_t/\alpha$. If $s = 3$ this cannot happen: z has a deadline strictly smaller than a job with weight M_t, necessarily the deadline of z is $t+2$, ADV schedules it at time $t+1$ when the job of weight M_t is still available, and z is not charged to z in EDF. Thus the proof for $s = 3$ is complete.

For the rest of the proof we renormalize so that $t = 0$ and $M_0 = \alpha$. Let z_i be the job scheduled by EDF at time i and let m be the smallest index such that $w_{z_m} = M_m$. (It exists, as eventually a maximal job is scheduled.) Thus $w_{z_i} = M_i/\alpha$ for $i < m$ and $\alpha = M_0 \leq M_1 \leq \cdots \leq M_m$. Let Z be the set of jobs $z_0, \ldots, z_{m-1}$.

Let $k \geq 0$ be such that $M_m = \alpha^{k+1}$. Let X_i, $i = 1, \ldots, k$, be the set of all jobs in Z with weight M_m/α^i that receive two charges. Let $X = \bigcup X_i$ be the set of all jobs in Z that receive two charges. Let $x_i = |X_i|$, $x = |X|$.

Using (I), each job in $Z - X$ is charged at most α times its weight. Let $W = \sum_{i=1}^{k} x_k/\alpha^i$, i.e., $WM_m = \alpha^{k+1}W$ is the total weight of jobs in X. Using

(II), the jobs in X and z_m are charged total at most $2WM_m + (1+1/\alpha)M_m$ and their weight is $WM_m + M_m$. To finish the proof of α-competitiveness, it is sufficient to show that

$$\alpha \geq \frac{2W+1+1/\alpha}{W+1} = 2 - \frac{1-\frac{1}{\alpha}}{W+1}.$$

The right-hand side increases with W, thus we need to determine the largest possible value of W.

Since the jobs in X receive also their own charge, each of them is scheduled in ADV and scheduled at time $m+1$ or later. By the definition of Z, EDF schedules first all jobs in X_k, then X_{k-1}, ..., up to X_1 (possibly some jobs in $Z - X$ are scheduled in between the jobs from X).

We claim that for each i,

$$x_i + x + 2 \leq s. \tag{8}$$

Let $i' \geq i$ be such that the last job finished by ADV from $X_i \cup \cdots \cup X_k$ belongs to $X_{i'}$. Let q be a job of maximal weight available when EDF schedules the first job of $X_{i'}$. Since q is scheduled by EDF only after all jobs in $X_{i'}$ despite the fact that its weight is larger, it must have strictly larger deadline than all the jobs in $X_{i'}$. Between the start of the first job in $X_{i'}$ in EDF and time $m+1$, EDF schedules all jobs in $X_{i'} \cup X_{i'-1} \cup \cdots \cup X_1$ and z_m, Between time $m+1$ and the time ADV finishes the last job of $X_{i'}$, ADV schedules all jobs in $X_i \cup \cdots \cup X_k$. Since the span of job q covers all of this and at least one time step after ADV finishes $X_{i'}$, we have

$$s \geq x_{i'} + x_{i'-1} + \cdots + x_1 + 1 + x_i + \cdots + x_k \geq x_i + x + 2.$$

Suppose that integers x_i satisfy (8) and maximize W. Then we claim that

- $x_i \geq x_{i+1}$ for any $i \geq 1$: otherwise switching the values of x_i and x_{i+1} is feasible and increases W;
- $x_i = 0$ for any $i \geq 3$: otherwise decreasing x_{i-1} and x_i both by 1 and increasing x_1 by 1 increases W by at least $1/\alpha - 1/\alpha^2 - 1/\alpha^3 \geq 0$, since $\alpha \geq \phi$.
- $2x_1 + x_2 = s - 2$: otherwise increasing x_1 by 1 and decreasing x_2 by 1 is feasible and increases W;

- $x_1 \leq x_2 + 2$: otherwise increasing x_2 by 2 and decreasing x_1 by 1 is feasible and increases W.

Thus the optimal values are $x_1 = \lfloor s/3 \rfloor$ and $x_2 = s - 2 - 2\lfloor s/3 \rfloor$ and for any case we have

$$2 - \frac{1 - \frac{1}{\alpha}}{W+1} \leq 2 - \frac{1 - \frac{1}{\alpha}}{1 + \frac{\lfloor \frac{s}{3} \rfloor}{\alpha} + \frac{s-2-2\lfloor \frac{s}{3} \rfloor}{\alpha^2}} = 2 - \frac{\alpha^2 - \alpha}{\alpha^2 + \lfloor \frac{s}{3} \rfloor \alpha + s - 2 - 2\lfloor \frac{s}{3} \rfloor} = \alpha$$

by the definition of α. □

B Appendix: Proof that $\textsc{Bal}_{\phi-1}$ is not ϕ-Competitive

Although we believe that $\textsc{Bal}_\beta$ is better than 2-competitive for some β, we can show that it cannot achieve competitive ratio ϕ. Even for 2-bounded instances the only values of β that can give ratio ϕ are $2-\phi$ and $\phi-1$, and it is not hard to give an example that $\textsc{Bal}_{2-\phi}$ is not ϕ-competitive in general. The hardest case is when $\beta = \phi - 1$ which we examine now.

Lemma B.1 *Let $p_1, \ldots, p_s$ be real numbers, such that $1 \leq p_1, p_2, \ldots, p_s \leq \phi + 1$. Let $P = \sum_{i=1}^{s} p_i$ and let $\epsilon > 0$ be arbitrarily small. Then there exists an instance (allowing negative time slots) such that at time 0, $\textsc{Bal}_{\phi-1}$ has pending exactly s jobs with weight p_i and deadline i, for $i = 1, \ldots, s$, and before time 0, $\textsc{Bal}_{\phi-1}$ has scheduled jobs of total weight $(\phi - 1)P - \epsilon$. In addition, there exists a schedule which schedules jobs with total weight $\phi \cdot P - \epsilon$ before time 0 (and has no jobs pending).*

Proof: Let $\{q_i\}_{i=1}^{s}$ be the numbers $p_1, \ldots, p_s$ ordered non-decreasingly. Consider (a long segment of) a backwards infinite sequence of jobs of span s, releasing one at each time step, with weights

$$\{q_1\phi^{-2n}, q_2\phi^{-2n}, \ldots, q_s\phi^{-2n}\}_{n=\infty}^{1},$$

arranged so that the last job, with weight $q_s\phi^{-2}$ is released at time $-s$ and has deadline 0. At time $-s+1$ we release a job of weight q_1 with deadline j such that $p_j = q_1$. At times $-s+2, \ldots, -1$ we continue with jobs of weights

$q_2, \ldots, q_s$ with appropriate deadlines so that at time 0, the s pending jobs have distinct deadlines and satisfy the requirements in the lemma.

At each time, a plan contains all pending jobs and the first job in the plan is tight. By the restriction on the weights q_i and their ordering, it follows that $\text{BAL}_{\phi-1}$ always schedules the planned job. The total scheduled weight is given by summing the infinite geometric sequence $P(\phi^{-2} + \phi^{-4} + \cdots) = P \cdot \phi^{-1} = (\phi - 1)P$. The adversary can schedule all the jobs as soon as they are released. □

Now we use the lemma to show that $\text{BAL}_{\phi-1}$ is not ϕ-competitive. The plan we start with, after applying the lemma, is

$$2, 2, \phi - \epsilon, \phi + 1, \phi + 1$$

for a small ϵ. Each job in the plan has a tight deadline. Note that $P = 3\phi + 6 - \epsilon$. The remaining jobs issued by the adversary are (in the (r, d, w) format):

$$\begin{aligned}
&(0, 2, 2), (0, 4, \phi + 1 + 10\epsilon),\\
&(1, 2, 2), (1, 5, \phi + 1 + 10\epsilon),\\
&(2, 3, \phi - 3\epsilon), (2, 4, 1 - \epsilon)
\end{aligned}$$

The adversary schedules always the first job in the pair upon its release and then the two jobs of weight $\phi+1+10\epsilon$. So its total gain (omitting $O(\epsilon) terms$) is:

$$adv = [\phi \cdot P] + [2 + 2 + \phi + (\phi + 1) + (\phi + 1)] = [9\phi + 3] + [3\phi + 6] = 12\phi + 9$$

$\text{BAL}_{\phi-1}$ schedules the second jobs of the first two pairs since

$$\begin{aligned}
\phi + 1 + 10\epsilon + (\phi - 1)(\phi - \epsilon) &= 10\epsilon + (\phi - 1)(-\epsilon) > \\
&\quad 10(\phi - 1)\epsilon = 2 + (\phi - 1)(\phi + 1 + 10\epsilon).
\end{aligned}$$

Then, at time 2, $\text{BAL}_{\phi-1}$ schedules the job with weight $\phi + 1$ and deadline 4, the comparison is

$$\phi + 1 + (\phi - 1)(1 - \epsilon) > \phi - \epsilon + (\phi - 1)(\phi + 1) = \phi - \epsilon + \phi.$$

Finally, at time 3, $\text{BAL}_{\phi-1}$ schedules the job with weight $\phi + 1$ and deadline 5, the comparison is

$$\phi + 1 > 1 - \epsilon + (\phi - 1)(\phi + 1) = 1 - \epsilon + \phi.$$

The total gain of $\text{BAL}_{\phi-1}$ is $gain = (\phi-1)P+4(\phi+1) = 3+6(\phi-1)+4(\phi+1) = 10\phi + 1$. Therefore we have $\phi \cdot gain = 11\phi + 10 < 12\phi + 9 = adv$, so $\text{BAL}_{\phi-1}$ is not ϕ-competitive.