# Online Competitive Algorithms for Maximizing Weighted Throughput of Unit Jobs

Yair Bartal[1], Francis Y. L. Chin[2], Marek Chrobak[3], Stanley P. Y. Fung[2], Wojciech Jawor[3], Ron Lavi[1], Jiří Sgall[4], and Tomáš Tichý[4]

[1] Institute of Computer Science, The Hebrew University of Jerusalem, Israel. yair,tron@cs.huji.ac.il
[2] Department of Computer Science and Information Systems, The University of Hong Kong, Hong Kong. chin,pyfung@csis.hku.hk
[3] Department of Computer Science, University of California, Riverside, CA 92521. marek,wojtek@cs.ucr.edu
[4] Mathematical Institute, AS CR, Žitná 25, CZ-11567 Praha 1, Czech Republic. sgall,tichy@math.cas.cz

**Abstract.** We study an online scheduling problem for unit-length jobs, where each job is specified by its release time, deadline, and a nonnegative weight. 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 $e/(e-1) \approx 1.582$. Then we consider $s$-bounded instances where the span of each job is at most $s$. We give a 1.25-competitive randomized algorithm for 2-bounded instances, and a deterministic algorithm $\textsc{Edf}_\alpha$, whose competitive ratio on $s$-bounded instances is at most $2 - 2/s + o(1/s)$. For 3-bounded instances its ratio is $\phi \approx 1.618$, matching the lower bound.
We also consider 2-uniform instances, where the span of each job is 2. We prove a lower bounds for randomized algorithms and deterministic memoryless algorithms. 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* [8, 1], packets arrive and are buffered at network switches. At each integer time step, one packet is 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 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, 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.

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. The goal is to compute a schedule for the given set of jobs that maximizes the *weighted throughput* or *gain*, that is the total weight of the jobs completed by their deadline.

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 [8, 1, 5]. 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 [8].

The unit-job scheduling problem is related to another scheduling problem which also arises from QoS applications. In *metered-task model* [2, 6], 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.

*Past work.* A naive greedy algorithm that always schedules the heaviest job is known to be 2-competitive [8, 7]. 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 [1, 5, 7]. In the randomized case, [5] give a lower bound of 1.25. (The proof in [5] 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 $\phi$-competitive algorithm was presented in [8]. Deterministic algorithms for 2-uniform instances were studied by [1], 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 [8], 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 [8] 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.

*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 [6].

For 2-bounded instances, we give a 1.25-competitive randomized algorithm, matching the known lower bound from [5].

We also give a deterministic algorithm $\textsc{Edf}_\alpha$ 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 [8], and it provides evidence that a $\phi$-competitive deterministic algorithm might be possible for the general case. For 4-bounded instances, $\textsc{Edf}_\alpha$ 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 [8]. 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 [1]. We remark that all competitive algorithms for unit-job scheduling in the literature are memoryless and scale-invariant.

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 $\textsc{Bal}_\beta$, where $\beta$ is a parameter, and we analyze it in several cases. On 2-uniform instances, $\textsc{Bal}_{\sqrt{2}/2}$ is $\sqrt{2}$-competitive, matching the bound from [1]. On 2-bounded instances, $\textsc{Bal}_\beta$ is $\phi$-competitive (and thus optimal) for two values of $\beta \in \{2 - \phi, \phi - 1\}$. It is also $\phi$-competitive for 3-bounded instances. Although we can show that $\textsc{Bal}_\beta$ cannot be $\phi$-competitive in general, we conjecture that for some values of $\beta$ 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.

## 2 Preliminaries

As we noted in the introduction, the QoS buffer management problem is equivalent to the unit-job scheduling problem. We will henceforth use 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. To simplify terminology and notation, we will

often use the weights of jobs to identify jobs. Thus, we will say "job $w$" meaning "the job with weight $w$". 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 $t$.

We often consider offline *(canonical) earliest-deadline* schedules. In such schedules, the job that is scheduled at any time $t$ is chosen (from the ones that are executed in the schedule) as the pending job with the earliest deadline. Any schedule can easily be converted into an earliest-deadline schedule by rearranging its jobs. Jobs with the same deadline are ordered by decreasing weights. (Jobs with equal weights are ordered arbitrarily, but consistently by all algorithms.)

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 $opt(I)/gain_{\mathcal{A}}(I)$. In most of the proofs we give a *potential function* argument by defining a potential function $\Phi$ 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.

**Lemma 1.** *Let $\mathcal{A}$ be an online algorithm. Let $\Phi$ 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 $\Delta gain_{\mathcal{A}}$, $\Delta adv$ 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.

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 [5] this model is called *non-timesharing*.) Then:

**Theorem 1.** *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 continuous version of the metered-task model [2, 6, 5] 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 the first randomized algorithm for scheduling 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 2.** *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 1, denoting by $j$ the job scheduled by the adversary, it sufficient to prove that $w_j + \Delta\Phi \leq \frac{e}{e-1}\omega$.

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

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

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 $\Phi$ 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 $w_j + \Delta\Phi \leq v_p + \sum_{i=1}^{p-1} \delta_i v_i = \frac{e}{e-1}(v_p - v_{k+1}) + \frac{1}{e-1}(v_1 - v_p) + \sum_{i=1}^{p-1} \delta_i v_i = \frac{e}{e-1}\sum_{i=p}^{k}(v_i - v_{i+1}) + \frac{1}{e-1}\sum_{i=1}^{p-1}(v_i - v_{i+1}) + \sum_{i=1}^{p-1} \delta_i v_i \leq \frac{e}{e-1}\sum_{i=p}^{k} \delta_i v_i + \frac{1}{e-1}\sum_{i=1}^{p-1} \delta_i v_i + \sum_{i=1}^{p-1} \delta_i v_i = \frac{e}{e-1}\omega$.

The easy case when $j \in Y - X$ is omitted.

## 4 An Optimal Randomized Algorithm for 2-Bounded Instances

In this section we give a 1.25-competitive randomized algorithm for 2-bounded instances. This matches the lower bound from [5], and thus completely resolves this case.

*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 3.** *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. 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 $a, b \geq 0$.

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

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

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 $\Phi_{xsz}$ 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 1, we need to show that for each adversary move:

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

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

Case 1: 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).

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

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

Case 2.1: 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 $2s'b + 5sp_{ab}a + 5tp_{x'b}x' = s(5p_{ab}a + 2q_{ab}b) + t(5p_{x'b}x' + 2q_{x'b}b) \geq 4sa + 4tx' \geq 4a$

Case 2.2: 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 $\phi$-competitive algorithm was given in [1], matching the lower bound from [8, 7]. In this section, we extend the upper bound of $\phi$ to 3-bounded instances. For the general case, the best known competitive ratio for deterministic algorithm is 2, [8, 7].

We define two algorithms. They both use a real-valued parameter $\alpha$ and $\beta$ and they are both $\phi$-competitive for 3-bounded instances for an appropriate value of the parameter. In this section, $h$ always denotes the heaviest pending job. The first algorithm schedules a relatively heavy job with the smallest deadline. 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. We conjecture that BAL is better than 2-competitive for general instances.

*Algorithm* $\text{EDF}_\alpha$*:* Execute the earliest-deadline job with weight $\geq \alpha w_h$.

*Algorithm* $\text{BAL}_\beta$*:* At each step, execute the job $j$ that maximizes $w_j + \beta\pi_j$, where $\pi_j$ 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 establish the following facts about $\text{BAL}_\beta$. All positive results can be shown using Lemma 1. In the following $x$ and $y$ are pending jobs of $\text{BAL}_\beta$ and the adversary, respectively.

- Let $\beta \in \{\phi - 1, 2 - \phi\}$, where $\phi \approx 1.618$ is the golden ratio. Then $\text{BAL}_\beta$ is $\phi$-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 $\Phi = \max(x, y) - \phi x$.
- $\text{BAL}_{\sqrt{2}/2}$ 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 $\Phi = (\sqrt{2} - 1)[y - 2x + \sqrt{2}(y - x)^+]$.
- $\text{BAL}_{\phi-1}$ is $\phi-$competitive for 3-bounded instances and is not $\phi-$competitive for 8-bounded instances. The proofs are omitted.

**Theorem 4.** $\text{EDF}_{\phi-1}$ *is $\phi$-competitive for 3-bounded instances.*

*Proof.* We fix a canonical earliest-deadline adversary schedule $A$. Let $E$ be the schedule computed by $\text{EDF}_{\phi-1}$. We use the following charging scheme: Let $j$ be 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 $t$.

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 $\text{EDF}_{\phi-1}$, $f$ 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 $\phi w_f$. 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 $\phi$-competitiveness of $\text{EDF}_{\phi-1}$ follows.

If $f$ receives only one charge, it is at most $\phi w_f$: 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 $\text{EDF}_{\phi-1}$. 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 $j$) But at time $t$, $\text{EDF}_{\phi-1}$ chooses $f$, so $j$ is not eligible for execution by $\text{EDF}_{\phi-1}$, 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 $\text{EDF}_{\phi-1}$, $d_h > d_f$. 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 $\text{EDF}_\alpha$ on $s$-bounded instances.

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

| $s$ | 2 | 3 | 4 | 5 | 6 | 7 | 10 | 20 | $\infty$ |
|---|---|---|---|---|---|---|---|---|---|
| $\text{EDF}_\alpha$ | 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 |

**Theorem 5.** *For each $s \geq 4$, algorithm $\text{EDF}_{1/\lambda_s}$ is $\lambda_s$-competitive for $s$-bounded instances, where $\lambda_s$ 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 1, results for discrete metered tasks can be applied to unit-job scheduling. Here we describe two such results. We say that 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 [4], 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. Thus, the results from [4] imply that:

**Theorem 6.** 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 $\text{EDF}_\alpha$ (with $\alpha = 1/\lambda_s$) and GAP. $\text{EDF}_\alpha$ has a smaller competitive ratio for $s$-bounded instances, but GAP can also be applied 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 [3], an algorithm FIT was given for the 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 $\xi$. By Theorem 1, we have:

**Theorem 7.** FIT *is $(2 - 1/(\lceil \lg \xi \rceil + 2))$-competitive for unit-job scheduling.*

## 6 2-Uniform Instances

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 [8].

**Theorem 8.** *No randomized algorithm can be better than $(4-2\sqrt{2})$-competitive for 2-uniform instances.*

*Proof.* We use Yao's minimax principle [9], 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 $j \leq n$, 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 $j \leq n$, 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 $j$. 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.

Deterministic algorithms for 2-uniform instances were studied in [1], where an upper bound of $\sqrt{2}$ was given. As we show below, it is not possible to beat ratio $\sqrt{2}$ with any deterministic memoryless, scale-invariant algorithm. We define an online algorithm $\mathcal{A}$ to be *memoryless* if its decision at each step depends only on the pending jobs. Due to space constraints the proof of the following theorem is omitted.

**Theorem 9.** *No deterministic memoryless algorithm can achieve competitive ratio better than $\sqrt{2}$ for 2-uniform instances.*

## 7 The Multiprocessor Case

The greedy 2-competitive algorithm [8, 7] applies to both uniprocessor and multiprocessor cases. For $M$ processors we give an algorithm with competitive ratio $(1-(\frac{M}{M+1})^M)^{-1}$, showing that the competitive ratio improves with a larger number of processors. When $M \to \infty$ this ratio tends to $e/(e-1) \approx 1.58$, beating the $\phi \approx 1.618$ bound for $M = 1$ [5]. The basic idea of our algorithm is similar to algorithm MIXED [5] and 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 time $t$. The algorithm chooses jobs $h_1, \ldots, h_M$ as shown below and schedules them for execution.

$i \leftarrow 1$;
`repeat`
    $g \leftarrow$ heaviest job in $X - \{h_1, ..., h_{i-1}\}$ ; $h_i \leftarrow g$ ; $j \leftarrow i$;
    `repeat`
        $i \leftarrow i + 1$;
        $f \leftarrow$ earliest-deadline job in $X - \{h_1, ..., h_{i-1}\}$ with $w_f \geq \beta^{i-j} w_g$;
        `if` $f$ exists `then` $h_i \leftarrow f$;
    `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 $h_k$, $1 < k \leq M$. By the choices of DMIX-$M$, we have $v_i \geq \beta^{i-1}$ for $i \in \{1, 2, ..., k-1\}$, $v_k < \beta^{k-1}$, and $v_i \geq v_k \beta^{i-k}$ for $i \in \{k+1, ..., M\}$.

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

**Theorem 10.** DMIX-$M$ *is* $(1 - (\frac{M}{M+1})^M)^{-1}$*-competitive for* $M$ *processors.*

*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 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 2, 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: The remaining case will be presented in the full version of the paper.

The lower bound proofs in [5] and Theorem 8 can easily be generalized to the multiprocessor case, improving the bounds in [8] ($4 - 2\sqrt{2}$ for the general case and 10/9 for the 2-uniform case):

**Theorem 11.** *No deterministic or randomized algorithm can be better than* 5/4*-competitive, for any number of processors $M$. No deterministic or randomized algorithm can be better than* $4-2\sqrt{2}$*-competitive, for* 2*-uniform instances on any number of processors $M$.*

**Acknowledgements** M. Chrobak and W. Jawor were supported by NSF grants CCR-9988360 and CCR-0208856. J. Sgall and T. Tichý were partially supported by Institute for Theoretical Computer Science, Prague (project LN00A056 of MŠMT ČR) and grant 201/01/1195 of GA ČR. M. Chrobak, W. Jawor, J. Sgall and T. Tichý were partially supported by cooperative grant KONTAKT-ME476/CCR-9988360-001 from MŠMT ČR and NSF. F. Y. L. Chin and S. P. Y. Fung were supported by an RGC research grant.

## References

1. 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.
2. E.-C. Chang and C. Yap. Competitive online scheduling with level of service. In *Proc. 7th COCOON*, pages 453–462. Springer, 2001.
3. F. Y. L. Chin and S. P. Y. Fung. Online scheduling with partial job values and bounded importance ratio. In *Proc. of ICS*, pages 787–794, 2002.
4. F. Y. L. Chin and S. P. Y. Fung. Improved competitive algorithms for online scheduling with partial job values. *9th COCOON, to appear*, 2003.
5. F. Y. L. Chin and S. P. Y. Fung. Online scheduling for partial job values: does timesharing or randomization help? *Algorithmica, to appear*, 2003.
6. 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.
7. B. Hajek. On the competitiveness of online scheduling of unit-length packets with hard deadlines in slotted time. In *CISS*, 2001.
8. 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.
9. A. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. of the 18th FOCS*, pages 222–227, 1977.