J Sched (2017) 20:623–634
DOI 10.1007/s10951-017-0522-4

Scheduling multipacket frames with frame deadlines

Łukasz Jeż[1] · **Yishay Mansour**[2,3] · **Boaz Patt-Shamir**[4]

Published online: 27 April 2017

Abstract We consider scheduling information units called frames, each with a delivery deadline. Frames consist of packets, which arrive online in a roughly periodic fashion, and compete on allocation of transmission slots. A frame is deemed useful only if all its packets are delivered before its deadline. We focus on a "proportional" variant, where the value of each frame is proportional to its size, but the sizes and periods of the frames are arbitrary. We give a constant-competitive algorithm for this setting, assuming bounded jitter and some slack in the frames' deadlines, the latter of which is necessary. Using standard techniques, our algorithm yields polylog-competitive algorithms for general instances with slack and bounded jitter.

Preliminary version of this article appeared in the Proceedings of 22nd International Colloquium on Structural Information and Communication Complexity (SIROCCO 2015). Łukasz Jeż was partially supported by the Polish National Science Center (NCN) Grant 2016/21/D/ST6/02402. Yishay Mansour was partially supported by the Israeli Centers of Research Excellence (I-CORE) program, Center No. 4/11, a Grant from the Israel Science Foundation (ISF), and a Grant from United States–Israel Binational Science Foundation (BSF). Boaz Patt-Shamir was partially supported by the Israel Science Foundation (Grant No. 1444/14) and by a Grant from Israel Ministry of Science and Technology. The authors would like to thank the anonymous referees for their comments.

✉ Łukasz Jeż
lje@cs.uni.wroc.pl

[1] Institute of Computer Science, University of Wrocław, ul. Joliot-Curie 15, 50-383 Wrocław, Poland

[2] Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel

[3] Microsoft Research, Herzlia, Israel

[4] School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel

Keywords Scheduling · Online algorithms · Communication networks

1 Introduction

In many networking settings, the flows entering the network have a nice periodic or almost periodic structure. The network would like to guarantee the flows a pre-specified quality of service (QoS), where one of the most basic QoS guarantees is a deadline by which the transfer would be completed. The uncertainty regarding the arrival of future flows motivates the online setting. We study this setting from the competitive analysis viewpoint. Let us start by giving a few motivating examples.

Consider a switch with multiple incoming video streaming flows competing for the same output link. Each flow consists of *frames*, and each frame consists of a variable number of *packets*. The video source is completely periodic, but due to compression, different frames may consist of a different number of packets. On top of that, asynchronous network transfer typically adds some jitter, so the input at the switch is only approximately periodic. In order for a frame to be useful, all its packets must be delivered before the frame's deadline. A frame is considered completed if all its packets are delivered before the frame's deadline, and the goal of a scheduling algorithm is to maximize the number of completed frames. Partially completed frames are considered worthless.

As another example, consider a Voice over IP (VoIP) setting. Voice calls generate samples at a relatively fast rate. Samples are wrapped in packets which are aggregated in logical frames with lower-granularity deadlines. Frames deadlines are more lax due to the tolerance of the human ear. Completed frames are reconstructed and replayed at the

receiver's side; incomplete frames are discarded, possibly resulting in an audible interruption (click) of the call. Our focus is on an oversubscribed link on the path of many such calls.

As a last example, consider a database or data center engaged in transferring truly huge files (e.g., petabytes of data) for replication purposes. It is common in such a scenario that the transfer must be completed by a certain given deadline. Typically, the transmission of such files is done piecemeal by breaking the file into smaller units, which are transmitted periodically so as to avoid overwhelming the network resources. We are interested in scenarios where multiple such transfers cross a common congested link.

Motivated by the above examples, we define the following abstract model. There are data units called *frames*, each with a *deadline* and a *value*. Each frame consists of several *packets*. Time is slotted. Packets arrive in an approximately periodic rate at a link, and can be transmitted (served) one packet at a step. A *scheduling algorithm* needs to decide which packet to transmit at each time slot. The goal of the algorithm is to maximize the total value of *delivered* frames, where a frame is considered delivered only if all its packets are transmitted before the frame's deadline.

The scheduling algorithm may be *committing* or *non-committing*. An algorithm is called committing if any packet it transmits belongs to a frame which is eventually delivered, whereas a non-committing algorithm may transmit a packet from some frame but later decide not to complete that frame.

Our performance measure is the competitive ratio, i.e., the worst-case ratio between the value delivered by the optimum (off-line) schedule and the online algorithm, over all arrival sequences.

Our Approach and Results Our model assumes that the arrival sequence is *not arbitrary*. Studying restricted instance classes and/or adversaries is common: Related work typically assumes specific order of frames and packets or restricted bursts. Instead, we assume that once the first packet of a frame arrives, and the arrival times of the remaining packets are predictable within a given bounded jitter. Moreover, we focus on an important natural special case of *proportional instances*, where the value of the frame is proportional to its size. The central conceptual contribution of this work is to demonstrate that there are deterministic algorithms with constant competitive ratio for proportional instances, if frames of very small periods are disallowed. For classic single machine scheduling, it is known (using substantially different techniques) how to obtain constant competitive ratio for proportional instances with bounded periods by a deterministic algorithm (Baruah et al. 1992). Choosing one of these two algorithms at random in the beginning of the execution yields a constant competitive *randomized* algorithm for all proportional instances. Moreover, let frame *density* be defined as the ratio of frame value to the number of its constituent packets. Then, the algorithm can be extended to general instances, with competitive ratio which is logarithmic in the ratio of maximum density to minimum density, cf. Sect. 4.

We also consider a special case of frames with common period and unit value but arbitrary sizes (and thus also densities). As already mentioned, there is a simple randomized algorithm whose competitive ratio is logarithmic in the maximum number of packets in a frame. We show that in this case a few natural algorithms, such as earliest deadline first (EDF) or shortest remaining processing time (SRPT), cannot guarantee a significantly better competitive ratio.

Related work The first multipacket-frame online model was introduced in Kesselman et al. (2013) and further studied in Scalosub et al. (2013). Emek et al. (2012) consider the basic model where the main difficulty is not deadlines but rather limited buffer space. Their results express the competitive ratio as a function of the maximum burst size and the number of packets in a frame. Subsequent work considered extensions to the basic model, including redundancy (Mansour et al. 2012), and hierarchically structured frames (Mansour et al. 2012; Scalosub et al. 2013).

Possibly the work closest to ours is Markovitch and Scalosub (2014), which essentially uses the same model, except that in Markovitch and Scalosub (2014), each *packet* has its deadline, and the packet arrivals may be arbitrary (whereas we assume that packets arrive approximately periodically). It is shown in Markovitch and Scalosub (2014) that the competitive ratio of the problem (both a lower and an upper bound) is exponential in the number of packets in a frame. One can view our results as showing that adding the extra assumptions that (1) the packet arrival is approximately periodic, i.e., with bounded jitter (and sufficient slack), (2) the value of each frame is proportional to its size, and that (3) packets belong to the same frame share the same deadline, allows for significantly improved competitive ratio, namely *constant* (or logarithmic if we drop assumption 2).

We note that the classic preemptive job scheduling problem of maximizing (weighted) throughput on a single machine (Kalyanasundaram and Pruhs 2000, 2003; Dürr et al. 2012; Lucier et al. 2013) corresponds to a special case of the problem we study in which all frames have period 1 and no jitter. For such setting, a combination of constant-competitive deterministic algorithm for proportional variant (Baruah et al. 1992) and the "classify and randomly select" technique (Awerbuch et al. 1994) yields an algorithm with competitive ratio logarithmic in the max-to-min ratio of frames densities. A matching (up to a constant factor) lower bound is known for randomized algorithms (Azar and Gilon 2015), and even stronger lower bounds (with focus on maxi-

mum job/frame size) apply to deterministic algorithms (Dürr et al. 2012). However, all these lower bounds rely on the presence of "tight" jobs, and constant-competitive algorithms, for instances, where all jobs have sufficient slack are known (Kalyanasundaram and Pruhs 2000; Lucier et al. 2013). We note that neither algorithm applies directly to our problem for the case of non-unitary periods.

Paper Organization We introduce the model and a few basic properties in Sect. 2. Our main result, i.e., for proportional instances, is given in Sect. 3. We point out how it extends to general instances (with a certain loss in the competitive ratio) in Sect. 4. Finally, in Sect. 5, we discus the case of different number of packets for each frame, assuming unit value and identical period.

2 Model and preliminary observations

In this section, we formalize the model and derive a few preliminary results. The notation we introduce here and use throughout the paper is summarized in Table 1.

We consider a standard scheduling model at the ingress of a link. Time is slotted, packets arrive online, and in each time slot at most one packet can be transmitted (meaning implicitly that we assume that all packets have the same length). The idiosyncrasies of our model are our assumptions about the arrival pattern and about the way the algorithm is rewarded for delivering packets, as described next.

Input: packets and frames The basic entities in our model are *frames* and *packets*. Each frame f has a *value* v_f and consists of $k_f \geq 1$ packets, where k_f is called its *size*. The value is a positive real number, and all other parameters (including the size) are nonnegative integers. We assume that packets of frame f arrive with *periodicity* $\pi_f \geq 1$ and *jitter* Δ_f, namely if packet 1 of f arrives at time $t_1(f) \in \mathbb{N}$, called the *frame f's arrival time*, then packet $i \in \{2, \ldots, k_f\}$ of f arrives at an integer *actual arrival time*

$$t_i(f) \in \left[\tau_i(f) - \Delta_f, \tau_i(f) + \Delta_f\right] , \tag{1}$$

where $\tau_i(f) \stackrel{\text{def}}{=} t_1(f) + (i-1)\pi_f$ is the packet i's *expected arrival time*. Each frame f has a *slack* $s_f \geq 1$,[1] which determines the *deadline* of f,

$$D_f \stackrel{\text{def}}{=} \tau_{k_f}(f) + \Delta_f + s_f .$$

The deadline D_p of a packet p of a frame f is simply D_f (see also the "output" paragraph below). A frame f is called *perfectly periodic* if $\Delta_f = 0$ and $s_f = \pi_f$. The parameters

[1] Our convention is that slack value of 1 means no slack. While somewhat unnatural, it helps reduce clutter later.

Table 1 Summary of frame-related notation

Symbol	Name/meaning/property
v_f	Value of frame f (if completed by deadline)
k_f	Size (number of packets) of frame f
ρ_f	Density of f; $\rho_f = v_f/k_f$
π_f	Period: packets of f arrive roughly every π_f steps
Δ_f	Jitter: max. abs. difference between $\tau_i(f)$ and $t_i(f)$
ι	Relative bound on jitter; $\forall_f \Delta_f \leq \iota\pi_f$
$\tau_i(f)$	Expected arrival time of i-th packet of f: $\tau_i(f) = t_1(f) + (i-1)\pi_f$
$t_i(f)$	Actual arrival time of i-th packet of f: $\lvert t_i(f) - \tau_i(f)\rvert \leq \Delta_f$, 0 for $i = 1$
$\tau(p), t(p)$	Expected and actual arrival time of packet p
D_f, D_p	Deadline of frame f and each of its packets p
s_f	Slack: $D_f - \left[t_1(f) + (k_f - 1)\pi_f + \Delta_f\right]$
σ	Relative bound on slack: $\forall_f s_f \geq \sigma\pi_f$
$p_{\min}$	Minimum over frames of parameter p_f
$p_{\max}$	Maximum over frames of parameter p_f

of a frame f (i.e., size k_f, value v_f, period π_f, jitter Δ_f and slack s_f) are made known to the algorithm when the first packet of f arrives; it is also convenient to introduce a (real-valued) frame's *density*, $\rho_f \stackrel{\text{def}}{=} v_f/k_f$. We denote the actual arrival time of the i-th packet of frame f, for $i \in \{1, \ldots, k_f\}$, by $t_i(f) \in \mathbb{N}$. The arrival time of the first packet of frame f, $t_1(f)$ is also called the arrival time of f. For convenience, we extend the arrival time notation to packets, i.e., for a given packet p, we let $\tau(p)$ and $t(p)$ denote p's expected and actual arrival time, respectively.

We assume that the algorithm knows nothing about a frame f before its arrival, and even then, it does not know the exact arrival times of the remaining packets, though it is guaranteed that they satisfy (1).

For a given instance, and a parameter $p \in \{\Delta, s, k, \pi, v, \rho\}$, we let $p_{\max} = \max_f(p_f)$ and $p_{\min} = \min_f(p_f)$, both taken over all frames in the instance, and extend these to instance classes. We assume that there exist constants $\iota, \sigma \geq 0$ such that $s_f \geq \sigma\pi_f$ and $\Delta_f \leq \iota\pi_f$ hold for all frames f. As we prove in Sect. 2.1, the first assumption is necessary, i.e., unless each frame's slack is (at least) proportional to its period, then no algorithm can attain constant competitive ratio. We also show that the slack has to be at least proportional to the jitter for the same reason. This motivates our second assumption: The relation we assume between jitter and period implies the required relation between jitter and slack, captures the idea of bounded jitter, and reduces the number of parameters. The notation for frame parameters is summarized in Table 1.

Output: delivered frames A *schedule* specifies which packet is transmitted in each slot (also called time step). A

slot is the interval $[t,\ t+1]$ spanned by two successive integer times t and $t+1$. The packet transmitted in the slot $[t,\ t+1]$ is *delivered* (or *completed*) at time $t+1$. A frame f is said to be *delivered* (or *completed*) in a given schedule if all its packets are transmitted at or before the frame deadline. The value of a given schedule is the sum of values of the frames that it completely delivers.

Algorithms The duty of an algorithm is to produce a schedule for any given arrival sequence, and the goal is to maximize the value of that schedule. An algorithm is called *online* if its decision at any time t depends only on the arrivals and transmissions at or before time t. Recall that we assume that all packet arrivals and completions occur at integer times, so any algorithm need make decisions only at such times. Specifically, the following sequence of events happens at time t: (1) the transmission of the packet that started at time $t-1$ (if any) completes, (2) all packets with actual arrival time t arrive, and (3) the algorithm chooses the packet to start transmitting at time t. We assume that the buffer space is unbounded, which means that the only contention is for the transmission slots.

We now define the notion of a *(non-)committing* algorithm: An algorithm is *committing* if it always completes a frame that it starts, i.e., has completed at least one packet of; such frame is called *accepted*; an algorithm without such property is called *non-committing*. We remark that a distinction between "commitment on arrival" and "commitment on admission" is sometimes made (Lucier et al. 2013): Our definition corresponds to commitment on admission, though our committing algorithm will in fact satisfy the stronger commitment on admission property.

The *competitive ratio* of an algorithm A is the worst-case ratio, over all arrival sequences α, between the maximal value delivered on α by any (off-line) schedule and the value delivered by A. Formally, the competitive ratio of A on instance class $\mathscr{I}$ is

$$\mathscr{R}_{\mathscr{I}}(A) \stackrel{\text{def}}{=} \sup_{\alpha\in\mathscr{I}} \frac{\text{OPT}(\alpha)}{A(\alpha)}\ ,$$

where $A(\alpha)$ denotes the value of the schedule produced by A on α, and $\text{OPT}(\alpha)$ denotes the maximal value of over all schedules for α. Note that $\mathscr{R}(A) \geq 1$ by definition.

2.1 Relation between slack, period, and jitter

Some settings of the parameters are uninteresting. In particular, we show that the following lower bound holds even if all frames have identical period, jitter, slack, value, and size.

Theorem 1 *No randomized algorithm on instances with all frames of size k, jitter Δ, slack s, and period π has competitive ratio smaller than*

$$\frac{\min\left\{2\Delta+s,\ \pi+\left\lfloor\frac{\Delta+s-\pi}{k}\right\rfloor\right\}}{s+\left\lfloor\frac{2\Delta}{k}\right\rfloor}\ ,$$

which is $\Omega\left(\min\{\Delta,\ \pi\}/s\right)$ for $k \gg \max\{\Delta, s, \pi\}$.

Before proving Theorem 1, we note that its bounds for large k motivate our assumptions that $s_f \geq \sigma\pi_f$ and $\Delta_f \leq \iota\pi_f$ for all frames f and some constants $\sigma, \iota > 0$. Note that the latter quantifies our assumption of "bounded jitter," and that both imply a bound $\Delta_f/s_f \leq \iota/\sigma$ for all frames f. Our main result, namely Theorem 2, can be seen as roughly matching these bounds in a far more general setting.

Proof (of Theorem 1) Consider the following scenario. At time 0, a large number n (determined later) of first packets of distinct frames arrive. These are the only frames in the instance. All their remaining packets, except the last ones, arrive exactly on time, i.e., at times $\pi, 2\pi, \ldots (k-2)\pi$. Let

$$t_0 \stackrel{\text{def}}{=} (k-1)\pi - \Delta,\ \text{and}$$
$$D \stackrel{\text{def}}{=} t_0 + 2\Delta + s$$

(t_0 is the earliest allowed arrival time of the last packet of the frames, and D is the deadline of all packets). For each frame f, let K_f be the random variable whose value is the number of f's packets that the algorithm delivers by the time t_0. Let F be the set of 2Δ frames with the lowest expected value of K_f, and let $\bar{F}$ denote the $n-2\Delta$ remaining frames. Note that $\sum_{f\in F}\mathbb{E}\left[K_f\right] \leq \frac{2t_0\Delta}{n}$. From time t_0 the arrivals continue as follows: All the last packets of frames of F arrive at time t_0, and all the last packets of frames in $\bar{F}$ arrive at time $t_0+2\Delta$. We now bound the number of frames completed by the algorithm. The number of frames from $\bar{F}$ completed by the algorithm is at most the number of last packets from $\bar{F}$ the algorithm sends before the deadline, which in turn is bounded by the number of time slots between their arrival and the deadline, i.e., s. In the interval $[t_0, t_0+2\Delta]$, the algorithm can complete only frames from F. Since only 2Δ packets can be sent in this interval, the expected number of completed frames from F is at most

$$\left\lfloor\frac{2\Delta+\sum_{f\in F}\mathbb{E}\left[K_f\right]}{k}\right\rfloor \leq \left\lfloor\frac{2\Delta}{k}\left(1+\frac{t_0}{n}\right)\right\rfloor \xrightarrow[n\to\infty]{} \left\lfloor\frac{2\Delta}{k}\right\rfloor\ ,$$

and hence, for n large enough, the expected number of frames completed by the algorithm is at most $s+\lfloor 2\Delta/k\rfloor$.

On the other hand, we now show that there is a schedule that delivers

$$x = \min\left\{2\Delta+s,\ \pi+\left\lfloor\frac{\Delta+s-\pi}{k}\right\rfloor\right\}$$

frames as follows. At time 0, choose the following set of frames X, with $|X| = x$. If $x \leq 2\Delta$, then X is any subset of F of size x, and if $x > 2\Delta$, X consists of F and $x-2\Delta$ arbitrary frames from $\bar{F}$. The schedule sends only packets of frames in X. We claim that all frames of X can be completed. To see this, consider the bipartite graph with a node for each packet of X and a node for each time slot in the interval $[0, D]$, and an edge connecting each packet-representing node p to all nodes representing a time-slot in which p can be sent, i.e., from its time of arrival until D. It is sufficient to show that in this graph there is a matching that matches all nodes representing packets of X. This in turn is easy to see using Hall's theorem. Let us number the packets of each frame 0 through $k-1$ by arrival order. Consider any subset P of packets of X, and let a_P be the first time slot in which a packet from P arrives. Suppose this is when a packet number i arrives. We proceed by case analysis according to the value of a_P. Note that $a_P \in \{i\pi \mid i = 0, \ldots, k-2\} \cup \{t_0, t_0 + 2\Delta\}$. If $a_P = i\pi$ for $i < k-1$, then $|P| \leq x(k-i) \leq \Delta + s + (k-i)\pi$ because $x \leq \pi + \left\lfloor \frac{\Delta+s-\pi}{k} \right\rfloor$, while the number of neighbors of P in the bipartite graph is $(k-i)\pi + \Delta + s \geq |P|$ and we are done in this case. If $a_p = t_0$, then P consists of last packets only, and hence, $|P| \leq x$, while the number of neighbors of P is $2\Delta + s \geq x$ and we are done in this case too. Finally, if $a_p = t_0 + 2\Delta$, then $P \subseteq X \setminus F$ and hence $|P| \leq s$, while the number of neighbors of P is s. □

We note that the bound of Theorem 1 is trivial for $s \geq \pi$, as in such case, the algorithm can easily complete (nearly) as many frames as OPT does.

2.2 Classify and randomly select

In this section, we briefly describe how the *classify and randomly select* technique (Awerbuch et al. 1994) can be applied to our problem.

Lemma 1 *Suppose that all frames can be partitioned into n classes, $C_1, \ldots, C_n$ such that, for each i, there is an algorithm A_i that is R_i-competitive on instances with frames of class C_i only. Then, the randomized algorithm $B(A_1, \ldots, A_n)$ (Algorithm 1) is $(\sum_j R_j)$-competitive.*

Algorithm 1 $B(A_1, \ldots, A_n)$:

1. Select $i \in \{1, \ldots, n\}$ with probability $p_i = R_i / \sum_j R_j$.
2. Run A_i on frames from C_i only, discarding packets of frames from other classes.

Proof Fix an instance I and suppose that a fixed optimum solution for I derives profit x_i from frames of class i. Then, for each i, the expected profit of the algorithm described above is at least $p_i x_i / R_i = x_i / \sum_j R_j$, since A_i is R_i-competitive. So overall, its expected profit is at least $\sum_j x_j / \sum_j R_j$. □

We note that the choice of probabilities in the randomized algorithm is optimal, as the relation between optimum profits x_i from different classes C_i are not known to the algorithm in advance and can be arbitrary. We also remark that Lemma 1 can be applied iteratively, i.e., a class C_i can be further partitioned into subclasses, e.g., based on a different parameter than the previous classification. A simple example of such iterative classification is found in Sect. 5.

3 Proportional instances

The main result of this section (and the whole article) is a constant-competitive randomized algorithm for proportional instances, which can be seen as almost matching the lower bound(s) of Theorem 1.

Throughout this section, we assume (w.l.o.g.) that the frame density is uniform, i.e., $\rho_f = v_f / k_f = 1$ for every frame f. Clearly, if the densities can range from $\rho_{\min}$ to $\rho_{\max}$, all the upper bounds we give still hold when scaled up by $r = \rho_{\max} / \rho_{\min}$. We assume that in proportional instances r is a constant, which we ignore. If a multiplicative factor of r is deemed too large, it can be reduced to $\mathcal{O}(\log r)$ using the classify and randomly select technique, cf. Sect. 4.

Theorem 2 *There exists an $\mathcal{O}\left(\frac{(1+\iota)^2}{\sigma}\right)$-competitive randomized algorithm for proportional instances.*

The algorithm in question is obtained by combining, as described in Sect. 2.2, two deterministic algorithms: LP (Algorithm 2) for frames of relatively large periods and SP (Algorithm 3) for frames of relatively small periods, presented and analyzed in the following two subsections. For convenience, we state the upper bounds on their competitive ratios here.

Theorem 3 *For instances with period at least $\pi_{\min} > \frac{1+\iota+\sigma}{\sigma}$, the Algorithm LP is $\left(\frac{2\pi_{\min}(1+\iota+\sigma)}{\sigma(\pi_{\min}-1)-(1+\iota)}+1\right)$-competitive.*

Theorem 4 *For instances with period at most $\pi_{\max}$, the Algorithm SP is $(4 + \pi_{\max} + \Delta_{\max})$-competitive.*

Proof (of Theorem 2) Denote the following algorithm as ALG(π). Frames of periods at most π form the class of small-period frames, and frames of periods at least $\pi + 1$ form the class of large-period frames. Then, $ALG(\pi)$ works as described in Sect. 2.2, using the algorithms SP (Algorithm 3) and LP (Algorithm 2) to handle frames of small and large periods, respectively. Lemma 1, Theorem 3 (with $\pi_{\min} = \pi +$

1), and Theorem 4 (with $\pi_{\max} = \pi$) yield that the competitive ratio of ALG(π) is at most

$$R(\pi) = \frac{2(\pi+1)(1+\iota+\sigma)}{\sigma\pi-(1+\iota)} + 5 + \pi(1+\iota) \ . \tag{2}$$

We now fix $\pi_0 = 2(1+\iota)/\sigma$, and prove that ALG ($\lceil\pi_0\rceil$) satisfies the theorem statement. To this end, we note that, in (2), the summand corresponding to the ratio of Algorithm LP is a decreasing function of π, and the one corresponding to the ratio of Algorithm SP is a linear function of π with slope $1+\iota$. Thus,

$$\begin{aligned} R(\lceil\pi_0\rceil) &\le R(\pi_0) + 1 + \iota \\ &\le \frac{2(\pi_0+1)(1+\iota+\sigma)}{1+\iota} + 5 + (\pi_0+1)(1+\iota) \\ &\le \frac{4\pi_0(1+\iota+\sigma)}{1+\iota} + 5 + 2\pi_0(1+\iota) \\ &= \frac{8(1+\iota+\sigma)}{\sigma} + 5 + \frac{4(1+\iota)^2}{\sigma} \ , \end{aligned}$$

which concludes the proof. □

We note that the $\mathcal{O}\left(\frac{(1+\iota)^2}{\sigma}\right)$ upper bound of Theorem 2 is the best (up to constant multiplicative factor) that can be derived from (2): As the denominator in the bound on LP's ratio has to be positive, $\pi > \frac{1+\iota}{\sigma}$ has to hold, but then SP's ratio is $\Omega\left(\pi(1+\iota)\right) = \Omega\left(\frac{(1+\iota)^2}{\sigma}\right)$.

3.1 Algorithms

3.1.1 Large periods

Our algorithm for instances with large periods, Algorithm LP (Algorithm 2), consists of two subroutines. The first decides, for each newly arriving frame, whether to accept or reject it, and the second schedules transmissions of packets of accepted frames. The algorithm classifies every frame as either *completed*, *accepted*, or *rejected*.

Algorithm 2 LP:

Frame Arrival: Accept a newly arriving frame f if and only if the set of all accepted frames together with f has a feasible schedule, assuming pessimal arrivals of all future packets of those frames.

Packet Transmission: Send the packet with the earliest deadline from the set of all pending packets of accepted frames. If the sent packet was the last one of its frame, mark the frame "completed."

The feasibility test that determines whether a frame is accepted considers packets rather than frames: The set of packets in question is that of all pending packets and those yet to arrive that belong either to an accepted frame or the frame f whose status is being decided. For the purpose of this test, the algorithm assumes that all such packets arrive as late as their frames' jitter allows. Clearly, this is the worst case for feasibility, i.e., if the set of frames is feasible under this assumption, it remains feasible if some packets arrive earlier than assumed. Testing the feasibility of a set of packets (which are just unit-length jobs) can be done by running EDF on that set, since EDF produces a (single machine) feasible schedule if there is one (Liu and Layland 1973). Conversely, our algorithm observes all deadlines because it produces an EDF schedule for a feasible set of packets.

Alternatively, the schedule for packets can be viewed as a *bipartite matching* of packets to time slots. Hence, one can test for feasibility with a new arriving frame f by using any dynamic matching algorithm that checks whether the current matching (schedule) can be augmented to match all packets of f as well. If so, the resulting schedule can then be reordered to become an EDF schedule.

We note that the algorithm either commits to a frame or rejects it immediately on its arrival. One can also consider a similar algorithm that commits on admission, i.e., decides to either accept or reject a frame only when its first packet would be scheduled by EDF. At such point, if the set of accepted frames together with f is feasible, then f is accepted and the packet is transmitted. Otherwise, f is rejected and another EDF packet is chosen for inspection. Intuitively, the algorithm that commits on admission should perform no worse than the one that does so on arrival. However, we consider the one that commits on arrival, for twofold reason: not only is commitment on arrival a more desirable property but also one that facilitates the analysis thanks to the algorithm's immediate decisions.

3.1.2 Small periods

Our algorithm, for instance, with small periods, Algorithm SP (Algorithm 3), is inspired by similar algorithms for proportional variants of job and interval (i.e., job with no slack) scheduling (Baruah et al. 1992; Woeginger 1994; Epstein et al. 2016). To introduce it, we define *critical time* of frame f as the latest time when f can still be completed if it was not scheduled at all before, i.e., $c_f = D_f - k_f$. Again, we present the algorithm as separate subroutines, handling frame arrivals, critical times, and transmissions.

3.2 Analyses

In the analyses, we often consider time intervals. For this reason, we now introduce some notation for those, which is mostly standard. For an interval I, we denote its length by $|I| \stackrel{\text{def}}{=} r(I) - l(I)$, where $l(I) \stackrel{\text{def}}{=} \inf I$ and $r(I) \stackrel{\text{def}}{=} \sup I$ are called the left and right endpoints of I, respectively.

Algorithm 3 SP:

State:

Q: a queue of frames, sorted by critical times; initially empty
f^a: *active frame*: a frame or $\perp$, initially $\perp$

Subroutines:

Frame Arrival: When a new frame f arrives, add it to Q.
Critical Time: When the critical time of the first frame in Q, f^q, is reached, it is removed from Q. Then, if $k_{f^q} \geq 2k_{f^a}$, preempt f^a and set $f^a := f^q$.
Transmission: If $f^a = \perp$ and Q is non-empty, set f^a to the first frame in Q and remove it from Q. Next, if there is a packet of f^a pending, send it. If this completes f^a, set f^a to $\perp$.

3.2.1 Large periods

To analyze Algorithm LP (Algorithm 2), we design a charging scheme. We charge a frame f delivered both by OPT and the algorithm to itself. Thus, we can restrict our attention to frames delivered by OPT that the algorithm rejected. To explain how these frames are charged, we need to introduce some notation.

We say that a set of time points *covers* a frame f if it contains a (closed) time interval $[a, b]$ such that $a \leq t_1(f)$ and $b \geq D_f$. A family of intervals $\mathscr{I}$ covers a frame f if its union $\bigcup_{I\in\mathscr{I}} I$ covers f. Moreover, for any family of intervals $\mathscr{I}$, we let $u(\mathscr{I}) = |\bigcup_{I\in\mathscr{I}} I|$ and $s(\mathscr{I}) = \sum_{I\in\mathscr{I}} |I|$.

We observe in Lemma 2 below that for every rejected frame f, there is an interval I_f that covers it such that the algorithm delivers a packet in roughly a constant fraction of I_f's slots; we call such I_f a *busy interval* and formalize this notion as follows.

Definition 1 Given a frame f, we say that an interval $I = [a, b]$ such that $a = t_1(f)$ and $b \geq D_f$ is *f-busy* if the algorithm delivers at least $\frac{\sigma}{\sigma+1+\iota}|I| - k_f$ packets within I, each with a deadline no larger than D_f.

Intuitively, the existence of an f-busy interval for every rejected frame f should yield a constant competitive ratio, for the following reasons. As the algorithm is committing and all the frames have the same density, we may count the transmitted packets rather than frames in the analysis. First, we observe that every frame f delivered by OPT that is not covered by a busy interval is delivered by the algorithm as well. And second, in each busy interval I, OPT can deliver at most $|I|$ packets, i.e., no more than a constant factor times the number that the algorithm does.

However, there are two issues. First, Lemma 2 states that the algorithm sends packets in $\frac{\sigma}{\sigma+1+\iota}|I_f| - k_f$ slots of a busy interval I_f, which means that we have a constant ratio on a packet basis only if I_f is sufficiently large. Fortunately, it follows from the lemma that short busy intervals correspond to rejected frames of small periods, which are handled by Algorithm SP (Algorithm 3).

Second, busy intervals may overlap, leading to overcounting of the packets delivered by the algorithm (and OPT). Thus, we need a claim similar to Lemma 2 for the union of all busy intervals. We remedy this by showing, in Lemma 3, that there is a family of disjoint busy intervals with total length that is at least half the one of the union of all the I_f intervals.

Lemma 2 *If Algorithm LP rejects a frame f_0, then there exists $T \geq D_{f_0}$ such that $[t_1(f_0), T]$ is f_0-busy.*

Proof Let B denote the set of all packets of f_0 and all unsent packets of frames that are accepted at time $t_1(f_0)$ (including packets that haven't arrived yet). Let $C = \{t \mid t \geq t_1(f_0)\}$ be the set of all time slots starting with $t_1(f_0)$.

We define a bipartite graph G whose nodes are $B \cup C$ as follows. For any $p \in B$, we define the *presumed arrival time of* p, denoted $t'(p)$, by

$$t'(p) = \begin{cases} t_1(f_0) & \text{if } t(p) \leq t_1(f_0) \\ \tau(p) + \Delta_{f(p)} & \text{otherwise} \end{cases}$$

where $f(p)$ is the frame that p belongs to (i.e., if p did not arrive yet, $t'(p)$ is the latest possible arrival time of p).

The edges of G connect each packet $p \in B$ with all time slots in which it may be sent assuming that it arrives at time $t'(p)$, i.e., p is connected to all time slots in the interval $\left[t'(p), D_{f(p)}\right]$.

Since f_0 was rejected, there is no matching of B to C in G with all nodes of B matched. Therefore, by Hall's theorem, there is a subset $P \subseteq B$ of packets such that $|P| > |\Gamma(P)|$, where $\Gamma(P)$ denotes the set of neighbors of P in G. We note that without loss of generality, $\Gamma(P)$ is an interval in C (i.e., a sequence of consecutive time slots): $\Gamma(P)$ is trivially a union of disjoint intervals in C. As the neighbors of each packet constitute an interval, the disjoint intervals of $\Gamma(P)$ induce a partition of P. Clearly, for at least one of these parts, say $P' \subseteq P$, we have that $|P'| > |\Gamma(P')|$. Thus, P' can be used instead of P, and we assume henceforth that $\Gamma(P) = [b, T]$. Since the set became infeasible only with the addition of f_0, it must be the case that P contains at least one packet of f_0, and hence, $T \geq D_{f_0}$. We shall show that $[t_1(f_0), T]$ is f_0-busy.

For all $a \in [t_1(f_0), b]$, let $P(a) \stackrel{\text{def}}{=} \left\{p \in P \mid t'(p) \geq a\right\}$. Let F denote the set of all frames f such that at least one packet of f belongs to P. Moreover, for every $f \in F$, let $n_f(a)$ denote the number of f's packets in $P(a)$ (i.e., $n_f(a) \stackrel{\text{def}}{=} |f \cap P(a)|$ if we view f as a set of packets) and $r_f(a) \stackrel{\text{def}}{=} n_f(a)/(T - a)$; we call the latter f's density in $[a, T]$. As mentioned before, since f_0 has been rejected, we have

$$\sum_{f\in F} r_f(b) = \sum_{f\in F} \frac{n_f(b)}{T-b} = \frac{1}{T-b}\sum_{f\in F} n_f(b) > 1 \ . \qquad (3)$$

To complete the proof, it suffices to prove that $|P(t_1(f_0))| > \frac{\sigma}{\sigma+1+\iota}|T-t_1(f_0)|$: As LP is a committing algorithm, it will complete at least this many packets in $[t_1(f_0),\ T]$ minus at most k_{f_0} packets of f_0, which belong to $P\ (t_1(f_0))$. Clearly, an even stronger inequality holds when $b = t_1(f_0)$ by (3), so assume that $b > t_1(f_0)$.

In such case, we give simple bounds for $n_f(a)$ and consequently also $r_f(a)$. Suppose that $n_f(t_f) \geq 1$ for some time t_f. Then, we know that for every decrease of the argument by π_f, n_f increases by 1, since π_f is the difference between presumed arrival times of two successive packets of f, with the following single exception: For the first packet of f, its "presumed arrival time" coincides with the actual arrival time, and thus precedes the presumed arrival time of the second packet by $\pi_f + \Delta_f$. Hence, for all $a \in \left[t_1(f_0), t_f\right]$:

$$\frac{\max\left\{t_f-\pi_f-\Delta_f-a,\ 0\right\}}{\pi_f} \leq n_f(a) - n_f(t_f) \leq \frac{t_f-a}{\pi_f} \ , \qquad (4)$$

which, used with $n_f\left(D_f - s_f\right) = 1$ and $D_f \leq T$, gives

$$r_f(a) = \frac{n_f(a)}{T-a} \leq \frac{n_f(a)}{D_f-a} \leq \frac{D_f - s_f + \pi_f - a}{\pi_f\left(D_f-a\right)} \ .$$

As both the numerator and the denominator are linear functions of a (or $D_f - a$), the bound is maximized for one the extreme arguments, i.e., either $D_f - a = s_f$ or $D_f - a \to \infty$, which yields

$$r_f(a) \leq \max\left\{\frac{1}{s_f},\ \frac{1}{\pi_f}\right\} \ . \qquad (5)$$

This allows us to lower bound the ratio of $r_f(a)$ to $r_f(b)$ by noting that $b \leq D_f - s_f$ for all $f \in F$, and hence, using (4), for all $a \in [t_1(f_0), b]$, we have

$$\begin{aligned}\frac{r_f(a)}{r_f(b)} &= \frac{n_f(a)}{(T-a)r_f(b)} \\ &\geq \frac{n_f(b) + \max\left\{\left\lceil\frac{b-a-\pi_f-\Delta_f}{\pi_f}\right\rceil,\ 0\right\}}{r_f(b)(T-a)} \ ,\end{aligned}$$

where the right-hand side is clearly minimized for $a = b - \Delta_f - x\pi_f$ for some positive integer x. Plugging this in, we get

$$\begin{aligned}\frac{r_f(a)}{r_f(b)} &\geq \min_{x\in\mathbb{N}_+} \frac{n_f(b)+x-1}{r_f(b)\left(T-b+\Delta_f+x\pi_f\right)} \\ &= \min_{x\in\mathbb{N}_+} \frac{(T-b)r_f(b)+x-1}{(T-b)r_f(b)+\left(\Delta_f+x\pi_f\right)r_f(b)} \ ,\end{aligned}$$

which, as both the numerator and the denominator are linear functions of x, is in turn minimized for either $x = 1$ or $x \to \infty$, yielding

$$\frac{r_f(a)}{r_f(b)} \geq \min\left\{\frac{T-b}{T-b+\pi_f+\Delta_f},\ \frac{1}{r_f(b)\pi_f}\right\} \ .$$

Now, we note that the left argument of the minimum is minimized for the minimum value of $T - b$, which is s_f. Therefore,

$$\begin{aligned}\frac{r_f(a)}{r_f(b)} &\geq \min\left\{\frac{s_f}{s_f+\pi_f+\Delta_f},\ \frac{1}{r_f(b)\pi_f}\right\} \\ &\geq \min\left\{\frac{\sigma}{\sigma+1+\iota},\ \frac{1}{r_f(b)\pi_f}\right\} \\ &\geq \min\left\{\frac{\sigma}{\sigma+1+\iota},\ \frac{\min\left\{s_f,\ \pi_f\right\}}{\pi_f}\right\} \\ &= \frac{\sigma}{\sigma+1+\iota} \ ,\end{aligned}$$

where the third inequality follows from (5) applied to the second argument of the minimum.

Thus, using above inequality and (3), $|P(a)|$ can be lower bounded as follows:

$$\begin{aligned}|P(a)| &= (T-a)\sum_{f\in F} r_f(a) = (T-a)\sum_{f\in F} r_f(b)\cdot\frac{r_f(a)}{r_f(b)} \\ &\geq (T-a)\inf_{f\in F}\frac{r_f(a)}{r_f(b)}\sum_{f\in F} r_f(b) \\ &> (T-a)\inf_{f\in F}\frac{r_f(a)}{r_f(b)} \geq \frac{\sigma}{\sigma+1+\iota}(T-a),\end{aligned}$$

which proves the lemma for $a = t_1\ (f_0)$. □

Next, we construct a family of busy intervals that underpins our analysis.

Lemma 3 *There exists a family $\mathscr{I}_0$ of closed intervals and its subset $\mathscr{I}_0' \subseteq \mathscr{I}_0$ that contains only disjoint intervals with the following properties:*

1. *Every $I \in \mathscr{I}_0$ is f-busy for some rejected frame f.*
2. *Every rejected frame is covered by $\mathscr{I}_0$.*
3. *$u\ (\mathscr{I}_0) \leq s\ (\mathscr{I}_0) \leq 2 \cdot u\ (\mathscr{I}_0)$.*
4. *$u\left(\mathscr{I}_0'\right) = s\left(\mathscr{I}_0'\right) \geq \frac{1}{2} \cdot s\ (\mathscr{I}_0)$.*

Proof We first show the construction of $\mathscr{I}_0$ and then that of $\mathscr{I}_0'$.

Let $\mathscr{I}$ contain an f-busy interval for every rejected frame f; existence of such intervals is guaranteed by Lemma 2.

We construct $\mathscr{I}_0$ by choosing a subset of $\mathscr{I}$, which ensures that the first property holds. To this end, we first partition $\mathscr{I}$ into components that correspond to the connected components in the interval graph of $\mathscr{I}$. (In such graph, the intervals from $\mathscr{I}$ form the vertices, and edges connect intervals that intersect.) To facilitate the description, for a family of time intervals $\mathscr{J}$ and a time point ξ, we let $\mathscr{J}^{\xi} = \{I \in \mathscr{J} \mid l(I) \leq \xi\}$. Now, for every component C, do the following. Initially, let $X_C = C$, $Y_C = \emptyset$, and $\xi = \min\{l(I) \mid I \in C\}$. Then, while $X_C \neq \emptyset$, do the following: Let $I \in X_C^{\xi}$ be the interval with the maximum $r(I)$; add I to Y_C, remove from X_C all I' such that $r(I') \leq r(I)$, and set $\xi := r(I)$. The family $\mathscr{I}_0$ is the union of all the (final) Y_C's over all components C, i.e., $\mathscr{I}_0 = \bigcup_C Y_C$.

By construction, every slot covered by $\mathscr{I}$ is covered by some $I \in \mathscr{I}_0$. Thus, the second property holds, as $\mathscr{I}$ covers every rejected frame f. (In fact, a stronger property holds for $\mathscr{I}$: each such f is covered by an interval $I \in \mathscr{I}$; this need not be the case for $\mathscr{I}_0$.)

To prove the third property, it suffices to restrict attention to a single component C as intervals of different components are disjoint. Let $I_1, I_2, \ldots$ be the intervals in Y_C in the order they were added to it. Note that both $l(I_i)$ and $r(I_i)$ strictly increase with i by construction. Moreover, as we consider intervals of one component, it holds that $l(I_{i+1}) \leq r(I_i)$, i.e., two successive intervals intersect. Now, consider three successive intervals I_i, I_{i+1}, I_{i+2} for any i. Then, $I_i \cap I_{i+2} = \emptyset$ because otherwise I_{i+2} would be a candidate for the successor of I_i and $r(I_{i+2}) > r(I_{i+1})$ would contradict the choice of I_{i+1}. This proves the second property as each slot in $\bigcup \mathscr{I}_0$ belongs to at most two intervals in $\mathscr{I}_0$.

Finally, we construct $\mathscr{I}_0'$ and prove that it satisfies the last property. To this end, from each Y_C, choose either all the odd-numbered or all the even-numbered intervals, whichever maximize the total length. This implies that $s(\mathscr{I}_0') \geq \frac{1}{2} s(\mathscr{I}_0)$. Moreover, it follows from the previous paragraph that the intervals in $\mathscr{I}_0'$ are disjoint. Thus, $u(\mathscr{I}_0') = s(\mathscr{I}_0')$ holds. □

Theorem 3 now follows from Lemmas 2 and 3.

Proof (of Theorem 3) The proof is via a charging scheme outlined in the beginning of this section. Let f be a frame that OPT delivered.

If the algorithm also delivered f, f is charged to itself. Clearly, each frame can receive at most one such charge.

All the frames rejected by the algorithm are charged simultaneously as follows. By Lemma 3, there is a family of busy intervals $\mathscr{I}_0$ that covers every frame rejected by the algorithm. In the time slots covered by $\mathscr{I}_0$, OPT delivers at most $u(\mathscr{I}_0) \leq s(\mathscr{I}_0)$ packets. A subset of these slots is covered by $\mathscr{I}_0'$, a family of disjoint busy intervals. It follows from the assumption and Lemma 2, that within each such busy interval $I_{f_0} \in \mathscr{I}_0'$ associated with a frame f_0, the algorithm delivers at least $\frac{\sigma}{\sigma+1+\iota}|I_{f_0}| - k_{f_0} \geq \left(\frac{\sigma}{\sigma+1+\iota} - \frac{1}{\pi_{\min}}\right)|I_{f_0}|$ packets. Thus, within all the time slots covered by $\mathscr{I}_0$, the algorithm delivers at least

$$\begin{aligned}
&\sum_{I \in \mathscr{I}_0'} \left(\frac{\sigma}{\sigma+1+\iota} - \frac{1}{\pi_{\min}}\right)|I| \\
&\geq \frac{1}{2}\left(\frac{\sigma}{\sigma+1+\iota} - \frac{1}{\pi_{\min}}\right) s(\mathscr{I}_0) \\
&= \frac{\sigma(\pi_{\min}-1)-(1+\iota)}{2\pi_{\min}(1+\iota+\sigma)} s(\mathscr{I}_0) \ ,
\end{aligned}$$

which proves the theorem. □

The downside of Theorem 3 is that it guarantees a constant-competitive algorithm only if $\pi_{\min}$ is bounded away from $(\sigma+1+\iota)/\sigma$, as can be seen in the final inequality of the proof. But precisely such frames can be handled by Algorithm SP (Algorithm 3).

3.2.2 Small periods

Proof (of Theorem 4) To prove the theorem, we focus a single phase, defined as follows. A new phase starts whenever in the transmission procedure the first condition holds, i.e., the algorithm removes a frame from the queue. Such phase lasts until the third condition holds (including that step), i.e., until the algorithm completes some active frame and sets f^a to $\perp$. Note that there may be gaps between phases when the algorithm has no active frame, and thus also no frames in its queue.

We denote the single phase we consider I, and associate it with the time interval in which it lasts. Let $f_0, f_1, \ldots, f_n$ denote all the successive active frames in I, and let $k_0, k_1, \ldots, k_n$ denote their respective sizes, i.e., $k_i = k_{f_i}$. We call the frames that were removed from the queue in I *considered*; note that this includes all the active frames, and that all the considered frames except possibly f_0 have reached their critical time. It is also easy to see that f_n has the largest deadline among all the considered frames, possibly except f_0, and that f_n completes exactly at its deadline if $n > 0$.

We note that I starts when the algorithms sends the first packet of f_0, and ends when it completes f_n, the only frame it completes in I. For this phase, we credit OPT with all of the considered frames that it completes plus the value of all the (other) packets that OPT sends in I; this one is a fractional value, i.e., credited per packet rather than per frame. We note that each considered frame has size less than $2k_n$ by the preemption rule. As each considered frame f, possibly excluding f_0, has reached its critical time within I, it follows that its deadline satisfies $D_f < r(I) + 2k_n$. Consequently, the gain of OPT associated with phase I is at most $|I| + 2k_n + k_0$.

Thus, it only remains to relate OPT to k_n. To this end, we note that, as all frames except f_0 have reached their critical time, and the sizes of active frames increase by at least a factor of 2:

$$\begin{aligned}\mathrm{OPT} &\le k_0 + |I| + 2k_n \\ &\le (k_0 - 1)\pi_{f_0} + \Delta_{f_0} + 1 + \sum_{i=0}^{n} k_i + 2k_n \\ &\le k_0\pi_{\max} + \Delta_{\max} + 4k_n \\ &\le k_n\,(4 + \pi_{\max} + \Delta_{\max})\ ,\end{aligned}$$

which concludes the proof, as every frame is considered in some phase. □

4 General instances

In this section we demonstrate how the results of Theorem 2 extend to general instances. As this is a well known application of the classify and randomly select technique (Awerbuch et al. 1994, see also Sect. 2.2), we keep the description brief.

Corollary 1 *There is an* $\mathcal{O}\left(\frac{(1+\iota)^2}{\sigma}\log\frac{\rho_{\max}}{\rho_{\min}}\right)$*-competitive randomized algorithm for general instances.*

Proof Partition the frames into $\left\lfloor \log\frac{\rho_{\max}}{\rho_{\min}}\right\rfloor+1$ classes by their density, so that the i-th class ($i \ge 0$) contains the frames of density $[2^i\rho_{\min}, 2^{i+1}\rho_{\min})$. Note that the densities in each class differ by at most a factor of 2. Thus, the (implicit) algorithm from (Theorem 2) guarantees a competitive ratio of $\mathcal{O}\left(\frac{(1+\iota)^2}{\sigma}\right)$ on each class exclusively, i.e., when the other classes are ignored. Hence, picking one of the classes uniformly at random in the beginning and ignoring frames of all other classes, yields an $\mathcal{O}\left(\frac{(1+\iota)^2}{\sigma}\log\frac{\rho_{\max}}{\rho_{\min}}\right)$-competitive algorithm by Lemma 1, cf. Sect. 2.2. □

5 Instances with common period and unit value

In this section we consider instances in which all frames have the same period π and unit value, but arbitrary sizes. Recall that such instances admit a randomized algorithm with competitive ratio $\mathcal{O}\left(\frac{(1+\iota)^2}{\sigma}\log\frac{k_{\max}}{k_{\min}}\right)$ by Corollary 1.

We could not find a better algorithm, even for perfectly periodic instances (to which Theorem 1 does not apply). In fact, two natural algorithms, EDF and SRPT, do not perform much better on those: we prove an $\Omega(\log k_{\max}/\log\log k_{\max})$ lower bound on their competitive ratios. We do not provide any upper bound for either of them. One could expect SRPT to be $\mathcal{O}(\log k_{\max})$-competitive, as it attains this ratio for single machine preemptive throughput maximization (Kalyanasundaram and Pruhs 2003; Dürr et al. 2012), which corresponds exactly to our setting with $\pi = 1$ and arbitrary s_f values. However, we do not know whether that analysis can be extended to our problem.

EDF and SRPT are defined as follows. At any given time t, we say that a frame f with deadline D_f is *feasible* if the number of remaining packets of f (ones that were not yet transmitted, including those that did not arrive yet) is at most $D_f - t$. Clearly, an infeasible frame cannot be delivered. At step t, both algorithms examine the set of all available packets of feasible frames, and transmits one chosen as follows. EDF chooses a packet of the frame with the earliest deadline. SRPT chooses a packet of the frame with the smallest number of remaining packets. Ties can be broken arbitrarily in both algorithms.

Since a frame's deadline is roughly its arrival time plus π times its size, these algorithms behave similarly. In particular, they share the following property: If the algorithm starts transmitting packets of a frame whose deadline is D, then by time D at least one frame is completed. However, ignoring long frames may not be the right choice, as the following theorem shows.

Theorem 5 *The competitive ratios of both EDF and SRPT on perfectly periodic instances with common period and unit values are* $\Omega\left(\frac{\log k_{\max}}{\log\log k_{\max}}\right)$.

Proof We construct an arrival sequence and inspect the number of frames delivered by either EDF or SRPT and by OPT. We first describe the construction parametrically.

Let $k_{\max}$ be given. Define $s_0 \stackrel{\text{def}}{=} 0$, $t_0 \stackrel{\text{def}}{=} \pi k_{\max}$, and $t_0' \stackrel{\text{def}}{=} t_0 - \varepsilon\pi k_{\max}$, where $\varepsilon \in (0, 1/2)$ is a parameter we fix later. For $i > 0$, define

$$\begin{aligned} t_i &\stackrel{\text{def}}{=} t_{i-1}' - \pi\ , \\ s_i &\stackrel{\text{def}}{=} t_i - \varepsilon^i\pi k_{\max}\ , \text{ and} \\ t_i' &\stackrel{\text{def}}{=} t_i - \varepsilon^{i+1}\pi k_{\max}\ . \end{aligned}$$

Let $H \stackrel{\text{def}}{=} \frac{\log k_{\max}}{\log(1/\varepsilon)}$. Note that $t_H - s_H = \varepsilon^H\pi k_{\max} = \pi$. The arrival sequence is as follows (see Fig. 1). At time s_i, for $i = 1, \ldots, H$, the first packets of frames from the following two sets arrive, c π frames.

- L_i are frames of $\varepsilon^i k_{\max}$ packets, i.e., their common deadline is at time $s_i + \varepsilon^i\pi k_{\max} = t_i$.
- S_i are frames of $\varepsilon^i(1-\varepsilon)k_{\max}$ packets, i.e., their common deadline is at time $s_i + \varepsilon^i(1-\varepsilon)\pi k_{\max} = t_i'$.

Let us now track the behavior of EDF and SRPT on this sequence (see Fig. 1)—we claim they both behave the same

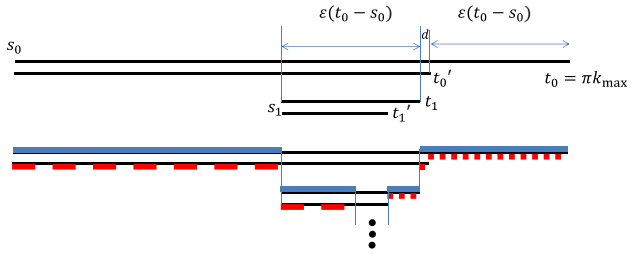

Fig. 1 Illustration of the top levels of the recursive construction. *Top*: representation of the start times and the deadlines of L_0, S_0, L_1, S_1. *Bottom*: representation of the optimal schedule (*blue solid lines*) and of EDF/SRPT (*red dashed lines*) (Color figure online)

way. At time 0, EDF (SRPT) starts working on packets from S_0, since they have the earliest deadline (the smallest size). At time s_1, EDF (SRPT) starts working exclusively on packets from S_1, because now they have the earliest deadline (the smallest size), and because they consume all bandwidth. In general, in the time interval $[s_i, s_{i+1})$, EDF (SRPT) sends only packets of frames from S_i. At time t'_H, EDF (SRPT) completes the π frames of S_H. These are the first frames completed by EDF (SRPT). Then, EDF (SRPT) starts processing the packets of L_H, which are now the ones with the earliest deadline (the smallest size), t_H. However, since EDF (SRPT) did not send any packet from L_H up to this time, and since $t_H - t'_H = \varepsilon^{H+1}\pi k_{\max}$, EDF (SRPT) completes only $\frac{\varepsilon^{H+1}\pi k_{\max}}{\varepsilon^H k_{\max}} = \varepsilon\pi$ frames from L_H. Then, in the interval $\left(t_H, t'_{H-1}\right]$, EDF (SRPT) can complete at most $\frac{\pi}{\varepsilon^H k_{\max}} = \pi$ frames from S_{H-1}, because each of the frames in S_{H-1} has $\varepsilon^H k_{\max} = 1$ unsent packets at time t_H.

This situation repeats in each level $0 \le i < H$. In the time interval $\left(t'_i, t_i\right]$ EDF (SRPT) transmits packets from L_i, and completes $\frac{t_i - t'_i}{\varepsilon^i k_{\max}} = \varepsilon\pi$ frames. In the time interval $\left(t_i, t'_{i-1}\right]$ EDF (SRPT) transmits packets from S_{i-1} and completes $\frac{t'_{i-1} - t_i}{\varepsilon^i k_{\max}} = \frac{\pi}{\varepsilon^i k_{\max}}$ frames, since each frame needs to complete the $\varepsilon^i k_0$ packets that arrive in $[s_i, t_i]$. Taking the π frames of S_H into account, the total number of frames sent by EDF (SRPT) in our scenario is

$$\begin{aligned}\text{EDF} &= \pi + \sum_{i=0}^{H-1} \pi \left(\varepsilon + \frac{1}{\varepsilon^i k_{\max}}\right) \\ &\le \pi + H\pi\varepsilon + \frac{\varepsilon^{-H}}{k_{\max}\,(1/\varepsilon - 1)}\pi \\ &= H\pi\varepsilon + \frac{\varepsilon}{1-\varepsilon}\pi \quad , \end{aligned} \tag{6}$$

because $\varepsilon^{-H} = k_{\max}$ by definition.

On the other hand, the optimal schedule is as follows. At the time interval $\left[s_i, s_{i+1}\right)$, only packets from L_i are sent, for $i = 0, \ldots H-1$. Then, at time t_H, all π frames from L_H are completed. Thereafter, at each interval $(t_{i+1}, t_i]$, only packets from L_i are sent. Now, a $(1-2\varepsilon)$-fraction of all packets of L_i were already sent in the interval $\left[s_i, s_{i+1}\right)$, and therefore each frame in L_i has only $2\varepsilon \cdot \varepsilon^i k_{\max}$ unsent packets remaining. Since $t_i - t_{i+1} = \varepsilon^{i+1}\pi k_{\max} + \pi$, the number of frames from L_i completed by the optimal schedule is $\left(\varepsilon^{i+1}\pi k_{\max} + \pi\right) / \left(2\varepsilon^{i+1} k_{\max}\right) > \pi/2$. It follows that the total optimal gain in our scenario is at least

$$\text{OPT} > \pi + \sum_{i=0}^{H-1} \pi/2 = \pi + H\pi/2\,. \tag{7}$$

Thus, the competitive ratio of EDF (SRPT) is at least

$$\frac{H\pi/2 + \pi}{H\pi\varepsilon + \pi(1-\varepsilon)^{-1}} \ge \frac{H/2}{H\varepsilon + (1-\varepsilon)^{-1}}\,.$$

Setting $\varepsilon = \frac{\log\log k_{\max}}{\log k_{\max}}$ implies $H\varepsilon = \Theta(1)$, as $H = \frac{\log k_{\max}}{\log(1/\varepsilon)}$. This, together with the fact that $\varepsilon < 1/2$, implies that the competitive ratio is $\Omega(H) = \Omega\left(\frac{\log k_{\max}}{\log\log k_{\max}}\right)$. □

We can show the following weak lower bound on the competitive ratio of randomized algorithms for common-period uniform-value instances.

Theorem 6 *Every randomized algorithm has competitive ratio at least* $4/3$ *on perfectly periodic instances with common period and uniform value.*

Proof Fix $k \gg k' \gg \pi$ and consider the following instance, in which π is the common period of all frames. At time 0, π *short frames* of size k and π *long frames* of size $k + k'$ arrive. No further frames arrive before time $(k-1)\pi$. Denote the expected number of packets of the short and long frames delivered by the algorithm by the time $(k-1)\pi$ by S and L, respectively. Clearly $S + L \le (k-1)\pi$, and hence, at least one of them is no larger than $(k-1)\pi/2$.

If $S \le (k-1)\pi/2$, then π frames of size 1 arrive at time $\left(k + k' - 1\right)\pi$. Otherwise (in which case $L \le (k-1)\pi/2$), π frames of size 1 arrive at time $(k-1)\pi$. Clearly, OPT can deliver 2π frames in the first case. In the second case, OPT delivers the π frames of size 1 in the interval $[(k-1)\pi, k\pi]$, and devotes all the time in $[0, (k+k')\pi]$ except aforementioned π slots to long frames. As at time $k\pi$ OPT has $k'\pi$ steps left and $k'+1$ packets of each long frame remaining, it can complete exactly $\frac{k'}{k'+1} \cdot \pi$ long frames. By our assumption that $k' \gg \pi$, this amounts to almost π long frames, and almost 2π frames in total.

As for the algorithm, we note that it cannot complete any long frame by time $k\pi$ since there are k' more packets yet to arrive for each long frame at that time. Similarly, it cannot complete any short frame after time $k\pi$ since it is their deadline. So in the first case, the expected number of

short frames that the algorithm can complete in the interval $[(k-1)\pi, k\pi]$ is at most $(S+\pi)/k \leq \pi(1+1/k)/2$, and then it can complete at most π frames in the interval $\left[\left(k+k'-1\right)\pi,\ \left(k+k'\right)\pi\right]$. By our assumption that $k \gg \pi$, the total (rounded down) number of completed frames in this case is at most $3\pi/2$. In the second case, the algorithm can complete at most π frames in the interval $[(k-1)\pi,\ k\pi]$, and the expected number of long frames it can then complete in the interval $\left[\left(k+k'-1\right)\pi, \left(k+k'\right)\pi\right)$ is at most $\left(L+k'\right)/\left(k+k'\right) \leq \pi \cdot \frac{k+2k'-1}{2(k+k')}$. Again, by our assumption that $k \gg k' \gg \pi$, the total (rounded down) number of completed frames in this case is at most $3\pi/2$. □

References

Awerbuch, B., Bartal, Y., Fiat, A., & Rosén, A. (1994). Competitive non-preemptive call control. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, (pp. 312–320). http://dl.acm.org/citation.cfm?id=314464.314510.

Azar, Y., & Gilon, O. (2015). Buffer management for packets with processing times. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA)*, (pp. 47–58). doi:10.1007/978-3-662-48350-3_5.

Baruah, S. K., Koren, G., Mao, D., Mishra, B., Raghunathan, A., Rosier, L. E., et al. (1992). On the competitiveness of on-line real-time task scheduling. *Real-Time Systems*, *4*(2), 125–144. doi:10.1007/BF00365406.

Dürr, C., Jeż, Ł., & Nguyen, K. T. (2012). Online scheduling of bounded length jobs to maximize throughput. *Journal of Scheduling*, *15*(5), 653–664. doi:10.1007/s10951-011-0233-1. Also appeared in *Proceedings of the 7th Workshop on Approximation and Online Algorithms (WAOA)*, (pp. 116–127). (2009).

Emek, Y., Halldórsson, M. M., Mansour, Y., Patt-Shamir, B., Radhakrishnan, J., & Rawitz, D. (2012). Online set packing. *SIAM Journal on Computing*, *41*(4), 728–746. doi:10.1137/110820774. Also appeared in *Proceedings of the 29th ACM Symposium on Principles of Distributed Computing (PODC)*, (pp. 440–449). (2010).

Epstein, L., Jeż, Ł., Sgall, J., & van Stee, R. (2016). Online scheduling of jobs with fixed start times on related machines. *Algorithmica*, *74*(1), 156–176. doi:10.1007/s00453-014-9940-2. Also appeared in *Proceedings of the 15th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, (pp. 134–145). (2012).

Kalyanasundaram, B., & Pruhs, K. (2000). Speed is as powerful as clairvoyance. *J. ACM*, *47*(4), 617–643. doi:10.1145/347476.347479. Also appeared in *Proc. of the 36th Symp. on Foundations of Comp. Sci.* (FOCS), (pp. 214–221). (1995).

Kalyanasundaram, B., & Pruhs, K. (2003). Maximizing job completions online. *Journal of Algorithms*, *49*(1), 63–85. doi:10.1016/S0196-6774(03)00074-9. Also appeared in *Proceedings of the 6th European Symposium on Algorithms (ESA)*, (pp. 235–246). (1998).

Kesselman, A., Patt-Shamir, B., & Scalosub, G. (2013). Competitive buffer management with packet dependencies. *Theoretical Computer Science*, *489–490*, 75–87. doi:10.1016/j.tcs.2013.04.007. Also appeared in *23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, (pp. 1–12). (2009).

Liu, C. L., & Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, *20*(1), 46–61. doi:10.1145/321738.321743.

Lucier, B., Menache, I., Naor, J., & Yaniv, J. (2013). Efficient online scheduling for deadline-sensitive jobs: Extended abstract. In *25th ACM Symposium on Parallelism in Algorithms and Architectures, (SPAA'13)*, (pp. 305–314). doi:10.1145/2486159.2486187.

Mansour, Y., Patt-Shamir, B., & Rawitz, D. (2012). Overflow management with multipart packets. *Computer Networks*, *56*(15), 3456–3467. doi:10.1016/j.comnet.2012.07.001. Also appeared in *Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM)*, (pp. 2606–2614). (2011).

Markovitch, M., & Scalosub, G. (2014). Bounded delay scheduling with packet dependencies. In *Proceedings of the IEEE INFOCOM Workshops*, (pp. 257–262). doi:10.1109/INFCOMW.2014.6849241.

Scalosub, G., Marbach, P., & Liebeherr, J. (2013). Buffer management for aggregated streaming data with packet dependencies. *IEEE Transactions on Parallel Distributed Systems*, *24*(3), 439–449. doi:10.1109/TPDS.2012.65. Also appeared in *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM)*, (pp. 241–245). (2010).

Woeginger, G. J. (1994). On-line scheduling of jobs with fixed start and end times. *Theoretical Computer Science*, *130*(1), 5–16. doi:10.1016/0304-3975(94)90150-3.