A 4/3-competitive randomized algorithm for online scheduling of packets with agreeable deadlines

Łukasz Jeż

Symposium on Theoretical Aspects of Computer Science 2010 (Nancy, France), pp. 489-500
www.stacs-conf.org

A $\frac{4}{3}$-COMPETITIVE RANDOMIZED ALGORITHM FOR ONLINE SCHEDULING OF PACKETS WITH AGREEABLE DEADLINES

ŁUKASZ JEŻ [1]

University of Wrocław, Institute of Computer Science
E-mail address: `lje@cs.uni.wroc.pl`
URL: `http://www.ii.uni.wroc.pl/ lje/`

ABSTRACT. In 2005 Li et al. gave a ϕ-competitive deterministic online algorithm for scheduling of packets with agreeable deadlines [12] with a very interesting analysis. This is known to be optimal due to a lower bound by Hajek [7]. We claim that the algorithm by Li et al. can be slightly simplified, while retaining its competitive ratio. Then we introduce randomness to the modified algorithm and argue that the competitive ratio against oblivious adversary is at most $\frac{4}{3}$. Note that this still leaves a gap between the best known lower bound of $\frac{5}{4}$ by Chin et al. [5] for randomized algorithms against oblivious adversary.

1. Introduction

We consider the problem of *buffer management with bounded delay* (aka *packet scheduling*), introduced by Kesselman et al. [11]. It models the behaviour of a single network switch. We assume that time is slotted and divided into steps. At the beginning of a time step, any number of packets may arrive at a switch and are stored in its *buffer*. A packet has a positive weight and a deadline, which is the number of step right before which the packet expires: unless it has already been transmitted, it is removed from the buffer at the very beginning of that step and thus can no longer be transmitted. Only one packet can be transmitted in a single step. The goal is to maximize the *weighted throughput*, i.e., the total weight of transmitted packets.

As the process of managing packet queue is inherently a real-time task, we investigate the online variant of the problem. This means that the algorithm has to base its decision of which packet to transmit solely on the packets which have already arrived at a switch, without the knowledge of the future.

1998 ACM Subject Classification: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems.

Key words and phrases: online algorithms, scheduling, buffer management.

This work has been supported by MNiSW grants no. N N206 1723 33, 2007–2010 and N N206 490638, 2010–2011.

1.1. Competitive Analysis.

To measure the performance of an online algorithm, we use a standard notion of competitive analysis [3], which compares the gain of the algorithm to the gain of the optimal solution on the same input sequence. For any algorithm ALG, we denote its gain on the input sequence I by $\mathcal{G}_{\text{ALG}}(I)$; we denote the optimal offline algorithm by OPT. We say that a deterministic algorithm ALG is $\mathcal{R}$-competitive if on any input sequence I, it holds that $\mathcal{G}_{\text{ALG}}(I) \geq \frac{1}{\mathcal{R}} \cdot \mathcal{G}_{\text{OPT}}(I)$.

When analysing the performance of an online algorithm ALG, we view the process as a game between ALG and an *adversary*. The adversary controls the packets' injection into the buffer and chooses which of them to send. The goal is then to show that the adversary's gain is at most $\mathcal{R}$ times ALG's gain.

If the algorithm is randomized, we consider its expected gain, $\mathbf{E}[\mathcal{G}_{\text{ALG}}(I)]$, where the expectation is taken over all possible random choices made by ALG. However, in the randomized case, the power of the adversary has to be further specified. Following Ben-David et al. [1], we distinguish between an *oblivious* and *adaptive-online* adversary (called adaptive for short). An oblivious adversary has to construct the whole input sequence in advance, not knowing the random bits used by an algorithm. The expected gain of ALG is compared to the gain of the optimal offline solution on I. An adaptive adversary decides packet injections upon seeing which packets are transmitted by the algorithm. However, it has to provide an answering entity ADV, which creates a solution on-line (in parallel to ALG) and cannot change it afterwards. We say that ALG is $\mathcal{R}$-competitive against an adaptive adversary if for any input sequence I created adaptively and any answering algorithm ADV, it holds that $\mathbf{E}[\mathcal{G}_{\text{ALG}}(I)] \geq \frac{1}{\mathcal{R}} \cdot \mathbf{E}[\mathcal{G}_{\text{ADV}}(I)]$. We note that ADV is (wlog) deterministic, but as ALG is randomized, so is the input sequence I.

In the literature on online algorithms (see e.g. [3]), the definition of the competitive ratio sometimes allows an additive constant, i.e., a deterministic algorithm is $\mathcal{R}$-competitive if there exists a constant $\alpha \geq 0$ such that $\mathcal{G}_{\text{ALG}}(I) \geq \frac{1}{\mathcal{R}} \cdot \mathcal{G}_{\text{OPT}}(I) - \alpha$ holds for evry input sequence I. An analogous definition applies to randomized case. Our upper bounds hold for $\alpha = 0$.

1.2. Previous work

The best known deterministic and randomized algorithms for general instances have competitive ratios at most $2\sqrt{2} - 1 \approx 1.828$ [6] and $e/(e-1) \approx 1.582$ [4], respectively. A recent analysis of the latter algorithm shows that it retains its competitive ratio even against adaptive-online adversary [8].

The best known lower bounds on competitive ratio against either adversary type use rather restricted 2-*bounded* sequences in which every packet has lifespan (deadline − release time) either 1 or 2. The lower bounds in question are $\phi \approx 1.618$ for deterministic algorithms [7], $\frac{4}{3}$ for randomized algorithms against adaptive adversary [2], and $\frac{5}{4}$ for randomized algorithms against oblivious adversary [5]. All these bounds are tight for 2-bounded sequences [11, 2, 4].

We restrict ourselves to sequences with *agreeable deadlines*, in which packets released later have deadlines at least as large as those released before ($r_i < r_j$ implies $d_i \leq d_j$). These strictly generalize the 2-bounded sequences. Sequences with agreeable deadlines also properly contain *s-uniform* sequences for all s, i.e., sequences in which every packet has lifespan exactly s. An optimal ϕ-competitive deterministic algorithm for sequences with agreeable deadlines is known [12].

Jeżabek studied the impact of *resource augmentation* on the deterministic competitive ratio [10, 9]. It turns out that while allowing the deterministic algorithm to transmit k packets in a single step for any constant k cannot make it 1-competitive (compared to the single-speed offline optimum) on unrestricted sequences [10], $k = 2$ is sufficient for sequences with agreeable deadlines [9].

1.3. Our contribution

Motivated by aforementioned results for sequences with agreeable deadlines, we investigate randomized algorithms for such instances. We devise a $\frac{4}{3}$-competitive randomized algorithm against oblivious adversary. The algorithm and its analysis are inspired by those by Li et al. [12] for deterministic case. The key insight is as follows. The algorithm MG by Li et al. [12] can be simplified by making it always send either e, the heaviest among the earliest non-dominated packets, or h, the earliest among the heaviest non-dominated packets. We call this algorithm MG′, and prove that it remains ϕ-competitive. Then we turn it into a randomized algorithm RG, simply by making it always transmit e with probability $\frac{w_e}{w_h}$ and h with the remaining probability. The proof of RG's $\frac{4}{3}$-competitiveness against oblivious adversary follows by similar analysis.

2. Preliminaries

We denote the release time, weight, and deadline of a packet j by r_j, w_j, and d_j, respectively. A packet j is *pending* at step t if $r_j \leq t$, it has not yet been transmitted, and $d_j > t$. We introduce a linear order $\trianglelefteq$ on the packets as follows: $i \trianglelefteq j$ if either

$$\begin{aligned} &d_i < d_j \ \text{, or} \\ &d_i = d_j \ \text{ and } w_i > w_j \ \text{, or} \\ &d_i = d_j \ \text{ and } w_i = w_j \ \text{ and } r_i \leq r_j \ . \end{aligned}$$

To make $\trianglelefteq$ truly linear we assume that in every single step the packets are released one after another rather then all at once, e.g. that they have unique fractional release times.

A *schedule* is a mapping from time steps to packets to be transmitted in those time steps. A schedule is *feasible* if it is injective and for every time step t the packet that t maps to is pending at t. It is convenient to view a feasible schedule S differently, for example as the set $\{S(t) \colon t > 0\}$, the sequence $S(1), S(2), \ldots$, or a matching in the *schedulability graph*. The schedulability graph is a bipartite graph, one of whose partition classes is the set of packets and the other is the set of time steps. Each packet j is connected precisely to each of the time steps t such that $r_j \leq t < d_j$ by an edge of weight w_j; an example is given in Figure 1. Observe that optimal offline schedules correspond to maximum weight matchings in the schedulability graph. Thus an optimal offline schedule can be found in polynomial time using the classic "Hungarian algorithm", see for example [13]. One may have to remove appropriately chosen time step vertices first, so that the remaining ones match the number of packet vertices, though.

Given any linear order $\preceq$ on packets and a (feasible) schedule S, we say that S is consistent with $\preceq$, or that S is a $\preceq$-schedule, if for every t the packet $S(t)$ is the minimum pending packet with respect to $\preceq$. It is fairly easy to observe that if $\preceq$ is any *earliest deadline*

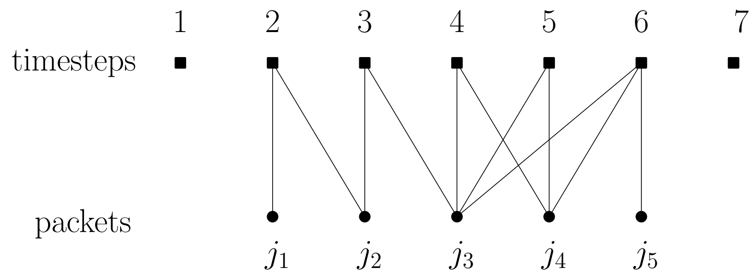


Figure 1: Schedulability graph for packets $j_1, j_2, \ldots, j_5$, whose release times and deadlines are (2,3), (2,4), (3,7), (4,7), (6,7) respectively; we ignore packet weights in the figure. Packets are represented by discs, time steps by squares.

first, with ties broken in an arbitrary way, then any feasible schedule can be turned to a unique $\preceq$-schedule by reordering its packets; in particular this applies to $\trianglelefteq$.

Recall that the *oblivious* adversary prepares the whole input sequence in advance and cannot alter it later on. Thus its solution is simply the offline optimal schedule for the complete sequence. Nevertheless, we still refer to the answering entity Adv rather than Opt in our analysis, as it involves altering the set of packets pending for the adversary, which may well be viewed as altering the input sequence. Now we introduce two schedules that are crucial for our algorithms and our analyzes.

Definition 2.1. The *oblivious schedule* at time step t, denoted O_t, is any fixed optimal feasible $\trianglelefteq$-schedule over all the packets pending at step t. For fixed O_t, a packet j pending at t is called *dominated* if $j \notin O_t$, and *non-dominated* otherwise. For fixed O_t let e denote $O_t(t)$, the $\trianglelefteq$-minimal of all non-dominated packets, and h denote the $\trianglelefteq$-minimal of all non-dominated maximum-weight packets.

Note that both the adversary and the algorithm can calculate their oblivious schedules at any step, and that these will coincide if their buffers are the same.

Definition 2.2. For a fixed input sequence, the *clairvoyant schedule* at time step t, denoted C_t, is any fixed optimal feasible schedule over all the packets pending at step t and all the packets that will arrive in the future.

Naturally, the adversary can calculate the clairvoyant schedule, as it knows the fixed input sequence, while the algorithm cannot, since it only knows the part of input revealed so far. However, the oblivious schedule gives some partial information about the clairvoyant schedule: intuitively, if p is dominated at t, it makes no sense to transmit it at t. Formally, (wlog) dominated packets are not included in the clairvoyant schedule, as stated in the following.

Fact 2.3. *For any fixed input sequence, time step t, and oblivious schedule O_t, there is a clairvoyant schedule C^*_t such that $C^*_t \cap \{j : r_j \leq t\} \subseteq O_t$.*

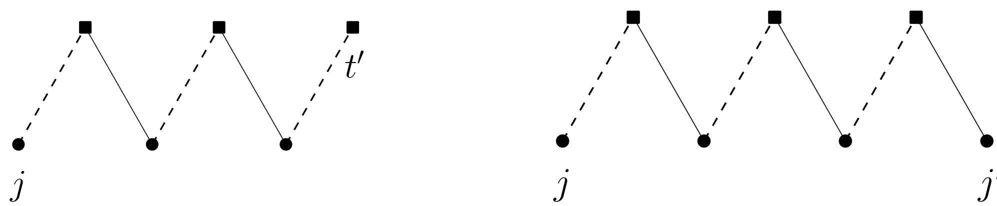
Proof. This is a standard alternating path argument about matchings. If you are unfamiliar with these concepts, refer to a book by A. Schrijver [13] for example.

Let O_t be the oblivious schedule and C_t be any clairvoyant schedule. Treat both as matchings in the schedulability graph and consider their symmetric difference $C_t \oplus O_t$.

Consider any job $j \in C_t \setminus O_t$ such that $r_j \leq t$. It is an endpoint of an alternating path P in $C_t \oplus O_t$. Note that all the jobs on P are already pending at time t: this is certainly true about j, and all the successive jobs belong to O_t, so they are pending as well.

First we prove that P has even length, i.e., it ends in a node corresponding to a job. Assume for contradiction that P's length is odd, and that P ends in a node corresponding to a timestep t'. Note that no job is assigned to t' in O_t. Then $O_t \oplus P$ is a feasible schedule that, treated as a set, satisfies $O_t \subseteq O_t \oplus P$ and $j \in O_t \oplus P$. This contradicts optimality of O_t. See Figure 2a for illustration.

Thus P has even length and ends with a job $j' \in O_t \setminus C_t$. By optimality of both O_t and C_t, $w_j = w_{j'}$ holds. Thus $C_t \oplus P$ is an optimal feasible schedule: in terms of sets the only difference between C_t and $C_t \oplus P$ is that j has been replaced by j', a job of the same weight. See Figure 2b for illustration.



(a) P cannot have odd length: in such case the assignment of jobs on P in O_t could be changed to match the strictly better assignment of C_t.

(b) P has even length: now the assignment of jobs on P in C_t can be changed to match the assignment of O_t so that the value of Δ drops.

Figure 2: The alternating path P. Packets are represented by discs, time steps by squares. Dashed lines represent C_t, solid lines represent O_t.

Applying such changes iteratively transforms C_t to a clairvoyant schedule C_t^* as announced. To observe that a finite number of iterations suffices, define $\Delta(S) := |S \cap \{j \colon r_j \leq t\} \setminus O_t|$ for any schedule S. It follows that $\Delta(C_t \oplus P) = \Delta(C_t) - 1$. Since Δ is non-negative and its value drops by one with each iteration, C_t^* is obtained in a finite number of steps. ∎

Definition 2.4. We say that a clairvoyant schedule C_t conforms with an oblivious schedule O_t if C_t is a $\trianglelefteq$-schedule, $C_t \cap \{j \colon r_j \leq t\} \subseteq O_t$, and for all $i \in O_t$ such that $i \triangleleft j = C_t(t)$, $w_i < w_j$ holds.

Fact 2.5. *For every oblivious schedule O_t there is a conforming clairvoyant schedule C_t^*.*

Proof. Let C_t be a clairvoyant schedule such that $C_t \cap \{j \colon r_j \leq t\} \subseteq O_t$; Fact 2.3 guarantees its existence. Let C_t^* be the schedule obtained from C_t by first turning it into a $\trianglelefteq$-schedule C_t' and then replacing $j = C_t'(t)$ with a $\trianglelefteq$-minimal non-dominated packet j' of the same weight.

If $j' = j$, then $C_t^* = C_t'$, and thus it is a clairvoyant $\trianglelefteq$-schedule. Assume $j' \neq j$, i.e., $j' \triangleleft j$. Then $j' \notin C_t'$, since C_t' is a $\trianglelefteq$-schedule. Thus C_t^* is feasible as we replace C_t''s very first packet by another pending packet which was not included in C_t. Observe that C_t^* is indeed a clairvoyant $\trianglelefteq$-schedule: optimality follows from $w_{j'} = w_j$, while consistency with $\trianglelefteq$ follows from $j' \triangleleft j$.

It remains to prove that for every $i \in O_t$ such that $i \lhd j'$, $w_i < w_{j'} = w_j$ holds. Note that $i \notin C_t^*$ as C_t^* is a $\unlhd$-schedule, and that $w_i \neq w_{j'} = w_j$ holds by the choice of j'. Assume for contradiction that $w_i > w_j$. Then C_t^* with j replaced by i is a feasible schedule contradicting optimality of C_t^*. ∎

Now we inspect some properties of conforming schedules.

Fact 2.6. *Let C_t be a clairvoyant schedule conforming with an oblivious schedule O_t. If $i, j \in O_t$, $w_i < w_j$ and $d_i < d_j$ (or, equivalently $w_i < w_j$ and $i \lhd j$), and $i \in C_t$, then also $j \in C_t$.*

Proof. Assume for contradiction that $j \notin C_t$. Then C_t with i replaced by j is a feasible schedule contradicting optimality of C_t. ∎

Lemma 2.7. *Let C_t be a clairvoyant schedule conforming with an oblivious schedule O_t. Suppose that $e = O_t(t) \notin C_t$. Then there is a clairvoyant schedule C_t^* obtained from C_t by reordering of packets such that $C_t^*(t) = h$.*

Proof. Let $j = C_t(t) \neq h$ and let $O_t = p_1, p_2, \ldots, p_s$. Observe that $h \in C_t$ by Fact 2.6. So in particular $e = p_1$, $j = p_k$, and $h = p_l$ for some $1 < k < l \leq s$. Let d_i denote the deadline of p_i for $1 \leq i \leq s$. Since O_t is feasible in the absence of future arrivals, $d_i \geq t + i$ for $i = 1, \ldots, s$.

Recall that $p_k, p_l \in C_t$ and that there can be some further packets $p \in C_t$ such that $p_k \lhd p \lhd p_l$; some of these packets may be not pending yet. We construct a schedule C_t' by reordering C_t. Precisely, we put all the packets from C_t that are not yet pending at t after all the packets from C_t that are already pending, keeping the order between the pending packets and between those not yet pending. By the agreeable deadlines property, this is an earliest deadline first order, so C_t' is a clairvoyant schedule.

As $e = p_1 \notin C_t'$ and $d_i \geq t + i$ for $i = 1, \ldots, s$, all the packets $x \in C_t'$ preceding h in C_t' (i.e., $x \in C_t'$ such that $r_x \leq t$ and $x \lhd p_l = h$) have *slack* in C_t', i.e., each of them could also be scheduled one step later. Hence $h = p_l$ can be moved to the very front of C_t' while keeping its feasibility, i.e., $C_t' = p_k, p_{k'}, \ldots, p_{l'}, p_l$ can be transformed to a clairvoyant schedule $C_t^* = p_l, p_k, p_{k'}, \ldots, p_{l'}$. The reordering is illustrated in Figure 3. ∎

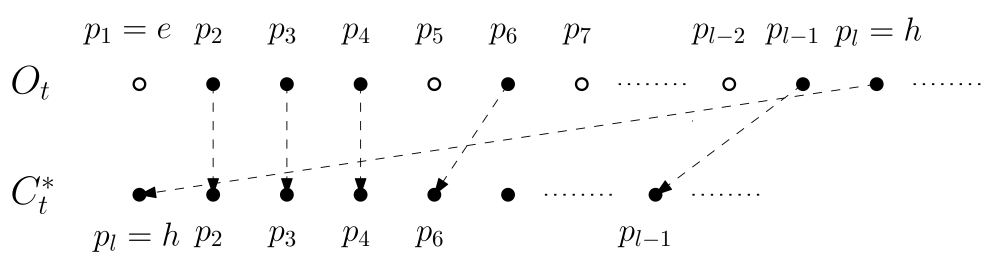


Figure 3: Construction of the schedule C_t^*. Packets are represented by circles: the ones included in C_t (C_t^*) are filled, the remaining ones are hollow.

3. Algorithms and their analyzes

3.1. The Algorithms

The algorithm MG [12] works as follows: at the beginning of each step t it considers the packets in the buffer and the newly arrived packets, and calculates O_t. Then MG identifies the packets e and h. If $\phi w_e \geq w_h$, MG sends e. Otherwise, it sends the $\trianglelefteq$-minimal packet f such that $w_f \geq \phi w_e$ and $\phi w_f \geq w_h$; the latter exists as h itself is a valid candidate. Our deterministic algorithm MG′ does exactly the same with one exception: if $\phi w_e < w_h$, it sends h rather than f. Our randomized algorithm RG also works in a similar fashion: it transmits e with probability $\frac{w_e}{w_h}$ and h with the remaining probability. For completeness, we provide pseudo-codes of all three algorithms in Figure 4.

MG (step t)

  $O_t \leftarrow$ oblivious schedule at t
  $e \leftarrow$ the $\trianglelefteq$-minimal packet from O_t
  $h \leftarrow$ the $\trianglelefteq$-minimal of all the heaviest packets from O_t
  if $\phi w_e \geq w_h$
    then transmit e
    else $f \leftarrow$ the $\trianglelefteq$-minimal of all $j \in O_t$ s.t. $w_j \geq \phi w_e$ and $\phi w_j \geq w_h$
      transmit f

MG′ (step t)

  $O_t \leftarrow$ oblivious schedule at t
  $e \leftarrow$ the $\trianglelefteq$-minimal packet from O_t
  $h \leftarrow$ the $\trianglelefteq$-minimal of all the heaviest packets from O_t
  if $\phi w_e \geq w_h$
    then transmit e
    else transmit h

RG (step t)

  $O_t \leftarrow$ oblivious schedule at t
  $e \leftarrow$ the $\trianglelefteq$-minimal packet from O_t
  $h \leftarrow$ the $\trianglelefteq$-minimal of all the heaviest packets from O_t
  transmit e with probability $\frac{w_e}{w_h}$ and h with probability $1 - \frac{w_e}{w_h}$

Figure 4: The three algorithms

3.2. Analysis Idea

The analysis of Li et al. [12] uses the following idea: in each step, after both MG and ADV transmitted their packets, modify ADV's buffer in such a way that it remains the same as MG's and that this change can only improve ADV's gain, both in this step and in the future. Sometimes ADV's schedule is also modified to achieve this goal, specifically, the packets in it may be reordered, and ADV may sometimes be allowed to transmit two packets in a single step. It is proved that in each such step the ratio of ADV's to MG's gain is at most ϕ. As was already noticed by Li et al. [12], this is essentially a potential function argument. To simplify the analysis, it is assumed (wlog) that ADV transmits its packets in the $\trianglelefteq$ order.

Our analysis follows the outline of the one by Li et al., but we make it more formal. Observe that there may be multiple clairvoyant schedules, and that ADV can transmit $C_t(t)$ at every step t, where C_t is a clairvoyant schedule chosen arbitrarily in step t. As our algorithms MG′ and RG determine the oblivious schedule O_t at each step, we assume that every C_t is a clairvoyant schedule conforming with O_t.

There is one exception though. Sometimes, when a reordering in C_t does not hinder ADV's performance (taking future arrivals into account), we "force" ADV to follow the reordered schedule. This is the situation described in Lemma 2.7: when $e \notin C_t$, there is a clairvoyant schedule C_t^* such that $h = C_t^*(t)$. In such case we may assume that ADV follows C_t^* rather than C_t, i.e., that it transmits h at t. Indeed, we make that assumption whenever our algorithm (either MG′ or RG) transmits h at such step: then ADV and MG′ (RG) transmit the same packet, which greatly simplifies the analysis.

Our analysis of MG′ is essentially the same as the original analysis of MG by Li et al. [12], but lacks one case which is superfluous due to our modification. As our algorithm MG′ always transmits either e or h, and the packet j that ADV transmits always satisfies $j \trianglelefteq h$ by definition of the clairvoyant schedule conforming with O_t, the case which MG transmits f such that $e \triangleleft f \triangleleft j$ does not occur to MG′. The same observation applies to RG, whose analysis also follows the ideas of Li et al.

3.3. Analysis of the Deterministic Algorithm

We analyze this algorithm as mentioned before, i.e., assuming (wlog) that at every step t ADV transmits $C_t(t)$, where C_t is a clairvoyant schedule conforming with O_t.

Theorem 3.1. MG′ *is ϕ-competitive on sequences with agreeable deadlines.*

Proof. Note that whenever MG' and ADV transmit the same packet, clearly their gains are the same, as are their buffers right after such step. In particular this happens when $e = h$ as then MG′ transmits $e = h$ and ADV does the same: in such case h is both the heaviest packet and the $\trianglelefteq$-minimal non-dominated packet, so $h = C_t(t)$ by definition of the clairvoyant schedule conforming with O_t.

In what follows we inspect the three remaining cases.

$\phi w_e \geq w_h$: MG′ transmits e. ADV transmits $j \neq e$.. To make the buffers of MG′ and ADV identical right after this step, we replace e in ADV's buffer by j. This is advantageous for ADV as $d_j \geq d_e$ and $w_j \geq w_e$ follows from $e \trianglelefteq j$ and the definition of a clairvoyant schedule conforming with O_t. As $\phi w_e \geq w_h$, the ratio of gains is

$$\frac{w_j}{w_e} \leq \frac{w_h}{w_e} \leq \phi \ .$$

$\phi w_e < w_h$: MG′ transmits h. ADV transmits e.. Note that ADV's clairvoyant schedule from this step contains h by Fact 2.6. We let ADV transmit both e and h in this step and keep e in its buffer, making it identical to the buffer of MG′. Keeping e, as well as transmitting two packets at a time is clearly advantageous for ADV. As $\phi w_e < w_h$, the ratio of gains is

$$\frac{w_e + w_h}{w_h} \leq \frac{1}{\phi} + 1 = \phi \ .$$

$\phi w_e < w_h$: MG′ transmits h. ADV transmits $j \neq e$.. Note that $j \trianglelefteq h$: by definition of the clairvoyant schedule conforming with O_t, for every $i \in O_t$ such that $i \triangleleft j$, $w_i < w_j$ holds.

There are two cases: either $j = h$, or $w_j < w_h$ and $d_j < d_h$. In the former one both players do the same and end up with identical buffers. Thus we focus on the latter case. Fact 2.6 implies that $h \in C_t$. By Lemma 2.7, C_t remains feasible when h is moved to its very beginning. Hence we assume that ADV transmits h in the current step. As this is the packet that MG′ sends, the gains of ADV and MG′ are the same and no changes need be made to ADV's buffer. ∎

3.4. Analysis of the Randomized Algorithm

We analyze this algorithm as mentioned before, i.e., assuming (wlog) that at every step t ADV transmits $C_t(t)$, where C_t is a clairvoyant schedule conforming with O_t.

Theorem 3.2. RG *is $\frac{4}{3}$-competitive against oblivious adversary on sequences with agreeable deadlines.*

Proof. Observe that if $e = h$, then RG transmits $e = h$ and ADV does the same: as in such case h is both the heaviest packet and the $\trianglelefteq$-minimal non-dominated packet, $h = C_t(t)$ by definition of the clairvoyant schedule conforming with O_t. In such case the gains of RG and ADV are clearly the same, as are their buffers right after step t. Thus we assume $e \neq h$ from now on.

Let us first bound the algorithm's expected gain in one step. It equals

$$\begin{aligned} \mathcal{G}_{\mathrm{RG}} &= \frac{w_e}{w_h} \cdot w_e + \left(1 - \frac{w_e}{w_h}\right) \cdot w_h \\ &= \frac{1}{w_h}\left(w_e^2 - w_e w_h + w_h^2\right) \\ &= \frac{1}{w_h}\left(\left(w_e - \frac{w_h}{2}\right)^2 + \frac{3}{4}w_h^2\right) \\ &\geq \frac{3}{4}w_h \ . \end{aligned} \tag{3.1}$$

Now we describe the changes to ADV's scheduling policy and buffer in the given step. These make ADV's RG's buffers identical, and, furthermore, make the expected gain of the adversary equal exactly w_h. This, together with (3.1) yields the desired bound. To this end we consider cases depending on ADV's choice.

(1) Adv transmits e. Note that Adv's clairvoyant schedule from this step contains h by Fact 2.6.

If RG transmits e, which it does with probability $\frac{w_e}{w_h}$, both players gain w_e and no changes are required.

Otherwise RG transmits h, and we let Adv transmit both e and h in this step and keep e in its buffer, making it identical to RG's buffer. Keeping e, as well as transmitting two packets at a time is clearly advantageous for Adv.

Thus in this case the adversary's expected gain is

$$\mathcal{G}_{\text{ADV}} = \frac{w_e}{w_h} \cdot w_e + \left(1 - \frac{w_e}{w_h}\right)(w_e + w_h) = w_e + (w_h - w_e) = w_h \ .$$

(2) Adv transmits $j \neq e$. Note that $j \trianglelefteq h$: by definition of the clairvoyant schedule conforming with O_t, for every $i \in O_t$ such that $i \lhd j$, $w_i < w_j$ holds.

If RG sends e, which it does with probability $\frac{w_e}{w_h}$, we simply replace e in Adv's buffer by j. This is advantageous for Adv as $w_j > w_e$ and $d_j > d_e$ follow from $e \lhd j$ and the definition of the clairvoyant schedule conforming with O_t.

Otherwise RG sends h, and we claim that (wlog) Adv does the same. Suppose that $j \neq h$, which implies that $w_j < w_h$ and $d_j < d_h$. Then $h \in C_t$, by Fact 2.6. Thus, by Lemma 2.7, C_t remains feasible when h is moved to its very beginning. Hence we assume that Adv transmits h in the current step. No further changes need be made to Adv's buffer as RG also sends h.

Thus in this case the adversary's expected gain is w_h. ■

4. Conclusion and Open Problems

We have shown that, as long as the adversary is oblivious, the ideas of Li et al. [12] can be applied to randomized algorithms, and devised a $\frac{4}{3}$-competitive algorithm this way. However, the gap between the $\frac{5}{4}$ lower bound and our $\frac{4}{3}$ upper bound remains.

Some parts of our analysis hold even in the adaptive adversary model [8]. On the other hand, other parts do not extend to adaptive adversary model, since in general such adversary's schedule is a random variable depending on the algorithm's random choices. Therefore it is not possible to assume that this "schedule" is ordered by deadlines, let alone perform reordering like the one in proof of Lemma 2.7.

This makes bridging either the $\left[\frac{5}{4}, \frac{4}{3}\right]$ gap in the oblivious adversary model, or the $\left[\frac{4}{3}, \frac{e}{e-1}\right]$ gap in the adaptive adversary model all the more interesting.

Acknowledgements

I would like to thank my brother, Artur Jeż, for numerous comments on the draft of this paper.

References

[1] S. Ben-David, A. Borodin, R. M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994. Also appeared in *Proc. of the 22nd STOC*, 1990.

[2] M. Bieńkowski, M. Chrobak, and Ł. Jeż. Randomized algorithms for Buffer Management with 2-Bounded Delay. In *Proc. of the 6th WAOA*, pages 92–104, 2008.

[3] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[4] F. Y. L. Chin, M. Chrobak, S. P. Y. Fung, W. Jawor, J. Sgall, and T. Tichý. Online competitive algorithms for maximizing weighted throughput of unit jobs. *J. Discrete Algorithms*, 4(2):255–276, 2006.

[5] F. Y. L. Chin and S. P. Y. Fung. Online scheduling with partial job values: Does timesharing or randomization help? *Algorithmica*, 37(3):149–164, 2003.

[6] M. Englert and M. Westermann. Considering suppressed packets improves buffer management in QoS switches. In *Proc. of the 18th SODA*, pages 209–218, 2007.

[7] B. Hajek. On the competitiveness of online scheduling of unit-length packets with hard deadlines in slotted time. In *Conference in Information Sciences and Systems*, pages 434–438, 2001.

[8] Ł. Jeż. Randomised buffer management with bounded delay against adaptive adversary. *CoRR*, abs/0907.2050, 2009.

[9] J. Jeżabek. Resource augmentation for QoS Buffer Management with Agreeable Deadlines. Unpublished, 2009^+.

[10] J. Jeżabek. Increasing machine speed in on-line scheduling of weighted unit-length jobs in slotted time. In *Proc. of the 35th SOFSEM*, pages 329–340, 2009.

[11] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. *SIAM J. Comput.*, 33(3):563–583, 2004.

[12] F. Li, J. Sethuraman, and C. Stein. An optimal online algorithm for packet scheduling with agreeable deadlines. In *Proc. of the 16th SODA*, pages 801–802, 2005.

[13] A. Schrijver. *Combinatorial Optimization*, volume A. Springer, 2003.