One to Rule Them All: A General Randomized Algorithm for Buffer Management with Bounded Delay

Łukasz Jeż[*]

August 28, 2018

Abstract

We give a memoryless scale-invariant randomized algorithm Mix-R for *buffer management with bounded delay* that is $e/(e-1)$-competitive against an adaptive adversary, together with better performance guarantees for many restricted variants, including the s-bounded instances. In particular, Mix-R attains the optimum competitive ratio of 4/3 on 2-bounded instances.

Both Mix-R and its analysis are applicable to a more general problem, called *Collecting Items*, in which only the relative order between packets' deadlines is known. Mix-R is the optimal *memoryless* randomized algorithm against adaptive adversary for that problem in a strong sense.

While some of provided upper bounds were already known, in general, they were attained by several different algorithms.

1 Introduction

In this paper, we consider the problem of *buffer management with bounded delay*, introduced by Kesselman et al. [16]. This problem models the behavior of a single network switch responsible for scheduling packet transmissions along an outgoing link as follows. We assume that time is divided into unit-length steps. At the beginning of a time step, any number of packets may arrive at a switch and be stored in its *buffer*. Each packet has a positive weight, corresponding to the packets priority, and a deadline, which specifies the latest time when the packet can be transmitted. Only one packet from the buffer can be transmitted in a single step. A packet is removed from the buffer upon transmission or expiration, i.e., reaching its deadline. The goal is to maximize the *gain*, defined as the total weight of the packets transmitted.

We note that *buffer management with bounded delay* is equivalent to a scheduling problem in which packets are represented as jobs of unit length, with given release times, deadlines and weights; release times and deadlines are restricted to integer values. In this setting, the goal is to maximize the total weight of jobs completed before their respective deadlines.

[*]University of Wrocław. Work supported by MNiSW grant N N206 490638, 2010–2011.

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

1.1 Competitive Analysis

To measure the performance of an online algorithm, we use the standard notion of *competitive analysis* [6], which, roughly speaking, compares the gain of the algorithm to the gain of the *optimal solution* on the same instance. For any algorithm $\textsc{Alg}$, we denote its gain on instance I by $\mathcal{G}_{\textsc{Alg}}(I)$. The optimal offline algorithm is denoted by $\textsc{Opt}$. We say that a deterministic algorithm $\textsc{Alg}$ is $\mathcal{R}$-competitive if on any instance I it holds that $\mathcal{G}_{\textsc{Alg}}(I) \geq \frac{1}{\mathcal{R}} \cdot \mathcal{G}_{\textsc{Opt}}(I)$.

When analyzing the performance of an online algorithm $\textsc{Alg}$, we view the process as a game between $\textsc{Alg}$ and an *adversary.* The adversary controls what packets are injected 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 $\textsc{Alg}$'s gain.

If the algorithm is randomized, we consider its expected gain, $\mathbb{E}[\mathcal{G}_{\textsc{Alg}}(I)]$, where the expectation is taken over all possible random choices made by $\textsc{Alg}$. However, in the randomized case, the power of the adversary has to be further specified. Following Ben-David et al. [3], we distinguish between an *oblivious* and an *adaptive-online* adversary, which from now on we will call *adaptive*, for short. An oblivious adversary has to construct the whole instance in advance. This instance may depend on $\textsc{Alg}$ but not on the random bits used by $\textsc{Alg}$ during the computation. The expected gain of $\textsc{Alg}$ is compared to the gain of the optimal offline solution on I. In contrast, in case of an adaptive adversary, the choice of packets to be injected into the buffer may depend on the algorithm's behavior up to the given time step. This adversary must also provide an answering entity $\textsc{Adv}$, which creates a solution in parallel to $\textsc{Alg}$. This solution may not be changed afterwards. We say that $\textsc{Alg}$ is $\mathcal{R}$-competitive against an adaptive adversary if for any adaptively created instance I and any answering algorithm $\textsc{Adv}$, it holds that $\mathbb{E}[\mathcal{G}_{\textsc{Alg}}(I)] \geq \frac{1}{\mathcal{R}} \cdot \mathbb{E}[\mathcal{G}_{\textsc{Adv}}(I)]$. We note that $\textsc{Adv}$ is (wlog) deterministic, but as $\textsc{Alg}$ is randomized, so is the instance I.

In the literature on online algorithms [6], the definition of the competitive ratio sometimes allows an additive constant, i.e., a deterministic algorithm is then called $\mathcal{R}$-competitive if there exists a constant $\alpha \geq 0$ such that for any instance I it holds that $\mathcal{G}_{\textsc{Alg}}(I) \geq \frac{1}{\mathcal{R}} \cdot \mathcal{G}_{\textsc{Opt}}(I) - \alpha$. An analogous definition applies to the randomized case. For our algorithm $\textsc{Mix-R}$ the bound holds for $\alpha = 0$, which is the best possible.

1.2 Basic Definitions

We denote a packet with *weight* w and *relative deadline* d by (w, d), where the relative deadline of a packet is, at any time, the number of steps after which it expires. The packet's *absolute deadline*, on the other hand, is the exact point in time at which the packet expires. a packet that is in the buffer, i.e., has already been released and has neither expired nor been transmitted by an algorithm, is called *pending* for the algorithm. The *lifespan* of a packet is its relative deadline value upon injection, or in other words the difference between its absolute deadline and release time.

The goal is to maximize the weighted throughput, i.e., the total weight of transmitted packets. We assume that time is slotted in the following way. We distinguish between points in time and time intervals, called *steps*. In step t, corresponding to the interval $(t, t+1)$, ADV and the algorithm choose, independently, a packet from their buffers and transmit it. The packet transmitted by the algorithm (ADV) is immediately removed from the buffer and no longer pending. Afterwards, at time $t+1$, the relative deadlines of all remaining packets are decremented by 1, and the packets whose relative deadlines reach 0 expire and are removed from both ADV's and the algorithm's buffers. Next, the adversary injects any set of packets. At this point, we proceed to step $t+1$.

To no surprise, all known algorithms are *scale-invariant*, which means that they make the same decisions if all the weights of packets in an instance are scaled by a positive constant. a class of further restricted algorithms is of special interest for their simplicity. An algorithm is *memoryless* if in every step its decision depends only on the set of packets pending at that step. An algorithm that is both memoryless and scale-invariant is called *memoryless scale-invariant*.

1.3 Previous and Related Work, Restricted Variants

The currently best, 1.828-competitive, deterministic algorithm for general instances was given by Englert and Westermann [10]. Their algorithm is scale-invariant, but it is *not* memoryless. However, in the same article Englert and Westermann provide another, 1.893-competitive, deterministic algorithm that is memoryless scale-invariant. The best known randomized algorithm is the 1.582-competitive memoryless scale-invariant RMIX, proposed by Chin et al. [7]. For reasons explained in Section 2.1 the original analysis by Chin et al. is only applicable in the oblivious adversary model. However, a refined analysis shows that the algorithm remains 1.582-competitive in the adaptive adversary model [14].

Consider a (memoryless scale-invariant) greedy algorithm that always transmits the heaviest pending packet. It is not hard to observe that it is 2-competitive, and actually no better than that. But for a few years no better deterministic algorithm for the general case was known. This naturally led to a study of many restricted variants. Below we present some of them, together with known results. The most relevant bounds known are summarized in Table 1. Note that the majority of algorithms are memoryless scale-invariant.

For a general overview of techniques and results on buffer management, see the surveys by Azar [2], Epstein and Van Stee [11] and Goldwasser [12].

Uniform Sequences An instance is *s-uniform* if the lifespan of each packet is exactly s. Such instances have been considered for two reasons. Firstly, there is a certain connection between them and the *FIFO model* of buffer management, also considered by Kesselmann et al. [16]. Secondly, the 2-uniform instances are among the most elementary restrictions that do not make the problem trivial. However, analyzing these sequences is not easy: while a simple deterministic 1.414-competitive algorithm for 2-uniform instances [18] is known to be optimal among memoryless scale-invariant algorithms [7], for unrestricted algorithms a sophisticated analysis shows the optimum competitive ratio is 1.377 [9].

Bounded Sequences An instance is *s-bounded* if the lifespan of each packet is at most s; therefore every s-uniform instances is also s-bounded. This class of in-

		deterministic	(rand.) adaptive	(rand.) oblivious
general	upper	1.828 [10], 1.893* [10]	**1.582*** [14]	1.582* [7]
	lower	1.618	1.333	1.25
s-bounded	upper	$2 - \frac{2}{s} + o(\frac{1}{s})^*$ [7]	$\mathbf{1/\left(1-(1-\frac{1}{s})^s\right)^*}$	$1/\left(1-(1-\frac{1}{s})^s\right)^*$
	lower	1.618	1.333	1.25
2-bounded	upper	1.618* [16]	**1.333*** [5]	1.25* [7]
	lower	1.618 [1, 8, 13]	1.333 [5]	1.25 [8]

Table 1: Comparison of known and new results. The results of this paper are shown in boldface; a reference next to such entry means that this particular bound was already known. The results without citations are implied by other entries of the table. An asterisk denotes that the algorithm attaining the bound is memoryless scale-invariant.

stances is important, because the strongest lower bounds on the competitive ratio known for the problem employ 2-bounded instances. These are $\phi \approx 1.618$ for deterministic algorithms [1, 8, 13], 1.25 for randomized algorithms in the oblivious adversary model [8], and 4/3 in the adaptive adversary model [5]. For 2-bounded instances algorithms matching these bounds are known [16, 7, 5]. A ϕ-competitive deterministic algorithm is also known for 3-bounded instances [7], but in general the best algorithms for s-bounded instances are only known to be $2 - 2/s + o(1/s)$-competitive [7].

Similarly Ordered Sequences An instance is *similarly ordered* or has *agreeable deadlines* if for every two packets i and j their spanning intervals are not properly contained in one another, i.e., if $r_i < r_j$ implies $d_i \leq d_j$. Note that every 2-bounded instance is similarly ordered, as is every s-uniform instance, for any s. An optimal deterministic ϕ-competitive algorithm [17] and a randomized 4/3-competitive algorithm for the oblivious adversary model [15] are known. With the exception of 3-bounded instances, this is the most general class of instances for which a ϕ-competitive deterministic algorithm is known.

Other restrictions Among other possible restrictions, let us mention one for which our algorithm provides some bounds. Motivated by certain transmission protocols, which usually specify only several priorities for packets, one might bound the number of different packet weights. In fact, Kesselmann et al. considered deterministic algorithms for instances with only two distinct packet weights [16].

Generalization: *Collecting Weighted Items from a Dynamic Queue* Bienkowski et al. [4] studied a generalization of buffer management with bounded delay, in which the algorithm knows only the relative order between packets' deadlines rather than their exact values; after Bienkowski et al. we dub the generalized problem *Collecting Items*. Their paper focuses on deterministic algorithms but it does provide certain lower bounds for memoryless algorithms, matched by our algorithm. See Appendix A for details.

1.4 Our Contribution

We consider randomized algorithms against an adaptive adversary, motivated by the following observation. In reality, traffic through a switch is not at all independent of the packet scheduling algorithm. For example, lost packets are typically resent, and throughput through a node affects the choice of routes for data streams in a network. These phenomena can be captured by the adaptive adversary model but not by the oblivious one. The adaptive adversary model is also of its own theoretical interest and has been studied in numerous other settings [6].

The main contribution of this paper is a simple memoryless scale-invariant algorithm Mix-R, which may be viewed as RMix, proposed by Chin et al. [7], with a different probability distribution over pending packets. The competitive ratio of Mix-R is at most $e/(e-1)$ on the one hand, but on the other it is provably better than that for many restricted variants of the problem. Some of the upper bounds we provide were known before (cf. Table 1), but in general they were achieved by several different algorithms.

Specifically, Mix-R is $1/\left(1-(1-\frac{1}{N})^N\right)$-competitive against adaptive adversary, where N is the maximum, over steps, number of packets that have positive probability of transmission in the step. Note that $1/\left(1-(1-\frac{1}{N})^N\right)$ tends to $e/(e-1)$ from below. The number N can be bounded a priori in certain restricted variants of the problem, thus giving better bounds for them, as we discuss in detail in Section 2.4. For now let us mention that $N \leq s$ in s-bounded instances and instances with at most s different packet weights. The particular upper bound of 4/3 that we obtain for 2-bounded instances is tight in the adaptive adversary model [5].

As is the case with RMix, both Mix-R and its analysis rely only on the relative order between the packets' deadlines. Therefore our upper bound(s) apply to the *Collecting Items* problem [4]. In fact, Mix-R is the optimum randomized memoryless algorithm for that problem in a strong sense, cf. Appendix A.

2 General Upper Bound

2.1 Analysis technique

In our analysis, we follow the paradigm of modifying the adversary's buffer, introduced by Li et al. [17]. Namely, we assume that in each step the algorithm and the adversary have precisely the same pending packets in their buffers. Once they both transmit a packet, we modify the adversary's buffer judiciously to make it identical with that of the algorithm. This amortized analysis technique leads to a streamlined and intuitive proof.

When modifying the buffer, we may have to let the adversary transmit another packet, inject an extra packet to his buffer, or upgrade one of the packets in its buffer by increasing its weight or deadline. We will ensure that these changes will be *advantageous to the adversary* in the following sense: for any adversary strategy Adv, starting with the current step and buffer content, there is an adversary strategy $\overline{\text{Adv}}$ that continues computation with the modified buffer, such that the total gain of $\overline{\text{Adv}}$ from the current step on (inclusive), on any instance, is at least as large as that of Adv.

To prove R-competitiveness, we show that in each step the expected *amortized gain* of the adversary is at most R times the expected gain of the algorithm, where the former is the total weight of the packets that ADV eventually transmitted in this step. Both expected values are taken over random choices of the algorithm.

We are going to assume that ADV never transmits a packet a if there is another pending packet b such that transmitting b is always advantageous to ADV. Formally, we introduce a dominance relation among the pending packets and assume that ADV never transmits a dominated packet.

We say that a packet $a = (w_a, d_a)$ is *dominated* by a packet $b = (w_b, d_b)$ at time t if at time t both a and b are pending, $w_a \leq w_b$ and $d_a \geq d_b$. If one of these inequalities is strict, we say that a is *strictly dominated* by b. We say that packet a is (strictly) dominated whenever there exists a packet b that (strictly) dominates it. Then the following fact can be shown by a standard exchange argument.

Fact 1. *For any adversary strategy* ADV*, there is a strategy* $\overline{\text{ADV}}$ *with the following properties:*

1. *the gain of* $\overline{\text{ADV}}$ *on every sequence is at least the gain of* ADV*,*
2. *in every step t,* $\overline{\text{ADV}}$ *does not transmit a strictly dominated packet at time t.*

Proof. ADV can be transformed into $\overline{\text{ADV}}$ iteratively: take the minimum t_0 such that ADV first violates the second property in step t_0, and transform ADV into an algorithm ADV$'$ with gain no smaller than that of ADV, which satisfies the second property up to step t_0, possibly violating it in further steps.

Let t_0 be the first step in which the second property is violated. Let $y = (w, d)$ be the packet transmitted by ADV and $x = (w', d')$ be the packet that dominates y; then $w' \geq w$ and $d' \leq d$. Let ADV$'$ transmit the same packets as ADV up to step $t_0 - 1$, but in step t_0 let it transmit x, and in the remaining steps let it try to transmit the same packets as ADV. It is impossible in one case only: when ADV transmits x in some step t. But then $d \geq d' > t$, so let ADV$'$ transmit y, still pending at t. Clearly, the gain of ADV$'$ is at least as large as the gain of ADV. □

Let us stress that Fact 1 holds for the adaptive adversary model. Now we give an example of another simplifying assumption, often assumed in the oblivious adversary model, which seems to break down in the adaptive adversary model. In the oblivious adversary model the instance is fixed in advance by the adversary, so ADV may precompute the optimum schedule to the instance and follow it. Moreover, by standard exchange argument for the *fixed* set of packets to be transmitted, ADV may always send the packet with the smallest deadline from that set—this is usually called the *earliest deadline first* (EDF) property or order. This assumption not only simplifies analyses of algorithms but is often crucial for them to yields desired bounds [7, 9, 17, 15].

In the adaptive adversary model, however, the following phenomenon occurs: as the instance I is randomized, ADV does not know for sure which packets it will transmit in the future. Consequently, deprived of that knowledge, it cannot ensure any specific order of packet transmissions.

2.2 The Algorithm

We describe the algorithm's behavior in a single step.

Algorithm 1 Mix-R (single step)

```
1: if there are no pending packets then
2:     do nothing and proceed to the next step
3: end if
4: $m \leftarrow 0$        ▷ counts packets that are not strictly dominated
5: $n \leftarrow 0$        ▷ counts packets with positive probability assigned
6: $r \leftarrow 1$        ▷ unassigned probability
7: $H_0 \leftarrow$ pending packets
8: $h_0 = (w_0, d_0) \leftarrow$ heaviest packet from $H_0$
9: while $H_m \neq \emptyset$ do
10:     $m \leftarrow m + 1$
11:     $h_m = (w_m, d_m) \leftarrow$ heaviest not strictly dominated packet from $H_{m-1}$
12:     $p_{m-1} \leftarrow \min\{1 - \frac{w_m}{w_{m-1}},\ r\}$
13:     $r \leftarrow r - p_{m-1}$
14:     if $r > 0$ then
15:         $n \leftarrow n + 1$
16:     end if
17:     $H_m \leftarrow \{x \in H_{m-1} \mid x$ is not dominated by $h_m\}$
18: end while
19: $p_m \leftarrow r$
20: transmit $h$ chosen from $h_1, \dots, h_n$ with probability distribution $p_1, \dots, p_n$
21: proceed to the next step
```

We introduce the packet h_0 to shorten Mix-R's pseudocode by making it possible to set the value of p_1 in the first iteration of the loop. The packet itself is chosen in such a way that $p_0 = 0$, to make it clear that it is not considered for transmission (unless $h_0 = h_1$). The while loop itself could be terminated as soon as $r = 0$, because afterwards Mix-R does not assign positive probability to any packet. However, letting it construct the whole sequence $h_1, h_2, \dots h_m$ such that $H_m = \emptyset$ simplifies our analysis. Before proceeding with the analysis, we note a few facts about Mix-R.

Fact 2. *The sequence of packets $h_0, h_1, \dots, h_m$ selected by* Mix-R *satisfies*

$$w_0 = w_1 > w_2 > \cdots > w_m \ ,$$
$$d_1 > d_2 > \cdots > d_m \ .$$

Furthermore, every pending packet is dominated by one of $h_1, \dots, h_m$.

Fact 3. *The numbers $p_1, p_2, \dots, p_m$ form a probability distribution such that*

$$p_i \leq 1 - \frac{w_{i+1}}{w_i} \qquad \text{for all } i < m \ . \tag{1}$$

Furthermore, the bound is tight for $i < n$, while $p_i = 0$ for $i > n$, i.e.,

$$p_i = \begin{cases} 1 - \frac{w_{i+1}}{w_i}, & \text{for } i < n \\ 0, & \text{for } i > n \end{cases} \tag{2}$$

Theorem 4. Mɪx-R *is* $1/\left(1-(1-\frac{1}{N})^N\right)$*-competitive against an adaptive adversary, where* N *is the maximum, over steps, number of packets that are assigned positive probability in a step.*

Proof. For a given step, we describe the changes to Aᴅᴠ's scheduling decisions and modifications to its buffer that make it the same as Mɪx-R's buffer. Then, to prove our claim, we will show that

$$\mathbb{E}\left[\mathcal{G}_{\text{Adv}}\right] \le w_1 \ , \tag{3}$$

$$\mathbb{E}\left[\mathcal{G}_{\text{Mix-R}}\right] \ge w_1\left(1-(1-\frac{1}{n})^n\right) \ , \tag{4}$$

where n is the number of packets assigned positive probability in the step. The theorem follows by summation over all steps.

Recall that, by Fact 1, Aᴅᴠ (wlog) sends a packet that is not strictly dominated. By Fact 2, the packets $h_1, h_2, \ldots h_m$ dominate all pending packets, so the one sent by Aᴅᴠ, say p is (wlog) one of $h_1, h_2, \ldots h_m$: if p is dominated by h_i, but not strictly dominated, then p has the same weight and deadline as h_i.

We begin by describing modifications to Aᴅᴠ's buffer and estimate Aᴅᴠ's amortized gain. To this end we need to fix the packet sent by Mɪx-R, so let us assume it is $h_f = (w_f, d_f)$. Assume that Aᴅᴠ transmits a packet $h_z = (w_z, d_z)$. We will denote the adversary's amortized gain given the latter assumption by $\mathcal{G}^{(z)}_{\text{Adv}}$. We consider two cases.

Case 1: $d_f \le d_z$. Then $w_f \le w_z$, since h_z is not dominated. After both Aᴅᴠ and Mɪx-R transmit their packets, we replace h_f in the buffer of Aᴅᴠ by a copy of h_z. This way their buffers remain the same afterwards, and the change is advantageous to Aᴅᴠ: this is essentially an upgrade of the packet h_f in its buffer, as both $d_f \le d_z$ and $w_f \le w_z$ hold.

Case 2: $d_f > d_z$. After both Aᴅᴠ and Mɪx-R transmit their packets, we let Aᴅᴠ additionally transmit h_f, and we inject a copy of h_z into its buffer, both of which are clearly advantageous to Aᴅᴠ. This makes the buffers of Aᴅᴠ and Mɪx-R identical afterwards.

We start by proving (3), the bound on the adversary's expected amortized gain. Note that Aᴅᴠ always gains w_z, and if $d_z < d_f$ ($z > f$), it additionally gains w_f. Thus, when Aᴅᴠ transmits h_z, its expected amortized gain is

$$\mathbb{E}\left[\mathcal{G}^{(z)}_{\text{Adv}}\right] = w_z + \sum_{i<z} p_i w_i \ . \tag{5}$$

As the adversary's expected amortized gain satisfies

$$\mathbb{E}\left[\mathcal{G}_{\text{Adv}}\right] \le \max_{1\le i\le m}\left\{\mathbb{E}\left[\mathcal{G}^{(i)}_{\text{Adv}}\right]\right\} \ ,$$

to establish (3), we will prove that

$$\max_{1\le i\le m}\left\{\mathbb{E}\left[\mathcal{G}^{(i)}_{\text{Adv}}\right]\right\} \le \mathcal{G}^{(1)}_{\text{Adv}} = w_1 \ . \tag{6}$$

The equality in (6) follows trivially from (5). To see that the inequality in (6) holds as well, observe that, by (5), for all $j < m$,

$$\mathbb{E}\left[\mathcal{G}^{(i)}_{\text{Adv}}\right] - \mathbb{E}\left[\mathcal{G}^{(i+1)}_{\text{Adv}}\right] = w_i - w_{i+1} - p_i w_i \ge 0 \ , \tag{7}$$

where the inequality follows from (1).

Now we turn to (4), the bound on the expected gain of Mix-R in a single step. Obviously,

$$\mathbb{E}\left[\mathcal{G}_{\text{Mix-R}}\right] = \sum_{i=1}^{n} p_i w_i \ . \tag{8}$$

By (2), $p_i w_i = w_i - w_{i+1}$ for all $i < n$. Also, $p_n = 1 - \sum_{i<n} p_i$, by Fact 3. Making corresponding substitutions in (8) yields

$$\begin{aligned}\mathbb{E}\left[\mathcal{G}_{\text{Mix-R}}\right] &= \left(\sum_{i=1}^{n-1}(w_i - w_{i+1})\right) + \left(1 - \sum_{i=1}^{n-1} p_i\right) w_n \\ &= w_1 - w_n \sum_{i=1}^{n-1} p_i \ . \end{aligned}\tag{9}$$

As (2) implies $w_i = w_{i-1}(1 - p_{i-1})$ for all $i \leq n$, we can express w_n as

$$w_n = w_1 \prod_{i=1}^{n-1}(1 - p_i) \ . \tag{10}$$

Substituting (10) for w_n in (9), we obtain

$$\mathbb{E}\left[\mathcal{G}_{\text{Mix-R}}\right] = w_1 \left(1 - \prod_{i=1}^{n-1}(1 - p_i) \sum_{i=1}^{n-1} p_i\right) \ . \tag{11}$$

Note that

$$\sum_{i=1}^{n-1}(1 - p_i) + \left(\sum_{i=1}^{n-1} p_i\right) = n - 1 \ ,$$

and therefore the inequality between arithmetic and geometric means yields

$$\prod_{i=1}^{n-1}(1 - p_i) \sum_{i=1}^{n-1} p_i \leq (1 - \frac{1}{n})^n \ . \tag{12}$$

Plugging (12) into (11) yields

$$\mathbb{E}\left[\mathcal{G}_{\text{Mix-R}}\right] \geq w_1 \left(1 - (1 - \frac{1}{n})^n\right) \ ,$$

which proves (4), and together with (3), the whole theorem. □

2.3 Rationale behind the probability distribution

Recall that the upper bound on the competitive ratio of Mix-R is

$$\frac{\max_{1 \leq z \leq m}\{\mathbb{E}\left[\mathcal{G}_{\text{Adv}}^{(z)}\right]\}}{\mathbb{E}\left[\mathcal{G}_{\text{Mix-R}}\right]} \ , \tag{13}$$

irrespective of the choice of $p_1, \ldots, p_m$.

The particular probability distribution used in Mix-R is chosen to (heuristically) minimize above ratio by maximizing $\mathbb{E}\left[\mathcal{G}_{\text{Mix-R}}\right]$, while keeping (6) satisfied, i.e., keeping $\mathbb{E}\left[\mathcal{G}_{\text{Adv}}\right] \le \mathcal{G}^{(1)}_{\text{Adv}} = w_1$.

The first goal is trivially achieved by setting $p_1 \leftarrow 1$. This however makes $\mathbb{E}\left[\mathcal{G}^{(z)}_{\text{Adv}}\right] > w_1$ for all $z > 1$. Therefore, some of the probability mass is transferred to $p_2, p_3, \ldots$ in the following way. To keep $\mathbb{E}\left[\mathcal{G}_{\text{Mix-R}}\right]$ as large as possible, p_2 is greedily set to its maximum, if there is any unassigned probability left, p_3 is set to its maximum, and so on. As $\mathbb{E}\left[\mathcal{G}^{(z)}_{\text{Adv}}\right]$ does not depend on p_i for $i \ge z$, the values $\mathbb{E}\left[\mathcal{G}^{(z)}_{\text{Adv}}\right]$ can be equalized with w_1 sequentially, with z increasing, until there is no unassigned probability left. Equalizing $\mathbb{E}\left[\mathcal{G}^{(j)}_{\text{Adv}}\right]$ with $\mathbb{E}\left[\mathcal{G}^{(j-1)}_{\text{Adv}}\right]$ consists in setting $p_{j-1} \leftarrow 1 - \frac{w_j}{w_{j-1}}$, as shown in (7). The same inequality shows what is intuitively clear: once there is no probability left and further values $\mathbb{E}\left[\mathcal{G}^{(z)}_{\text{Adv}}\right]$ cannot be equalized, they are only smaller than w_1.

The lower bound for the *Collecting Items* problem [4], presented in Appendix A, proves that this heuristic does minimize (13).

2.4 Implications for Restricted Variants

We have already mentioned that for s-bounded instances or those with at most s different packet weights, $N \le m \le s$ in Theorem 4, which trivially follows from Fact 2. Thus for either kind of instances Mix-R is $1/\left(1 - (1 - \frac{1}{s})^s\right)$-competitive. In particular, on 2-bounded instances Mix-R coincides with the previously known optimal 4/3-competitive algorithm Rand [5] for the adaptive adversary model.

Sometimes it may be possible to give more sophisticated bounds on N, and consequently on the competitive ratio for particular variant of the problem, as we now explain. The reason for considering only the packets $h_0, h_1, \ldots, h_m$ is clear: by Fact 1 and Fact 2, Adv (wlog) transmits one of them. Therefore, Mix-R tries to mimic Adv's behavior by adopting a probability distribution over these packets (recall that in the analysis the packets pending for Mix-R and Adv are exactly the same) that keeps the maximum, over Adv's choices, expected amortized gain of Adv and its own expected gain as close as possible (cf. Section 2.3). Now, if for whatever reason we know that Adv is going to transmit a packet from some set S, then H_0 can be initialized to S rather than all pending packets, and Theorem 4 will still hold. And as the upper bound guaranteed by Theorem 4 depends on N, it might improve if the cardinality of S is small.

While it seems unlikely that bounds for any restricted variant other than s-bounded instances or instances with at most s different packet weights can be obtained this way, there is one interesting example that shows it is possible. For similarly ordered instances (aka instances with agreeable deadlines) and oblivious adversary one can always find such set S of cardinality at most 2 [15, Lemma 2.7]; while not explicitly stated, this fact was proved before by Li et al. [17]. Roughly, the set S contains the earliest-deadline and the heaviest packet from any optimal provisional schedule. The latter is the optimal schedule under the assumption that no further packets are ever injected, and as such can be found in any step.

3 Conclusion and Open Problems

While Mix-R is very simple to analyze, it subsumes almost all previously known randomized algorithms for packet scheduling and provides new bounds for several restricted variants of the problem. One notable exception is the optimum algorithm against oblivious adversary for 2-bounded instances [7]. This exposes that the strength of our analysis, i.e., applicability to adaptive adversary model, is most likely a weakness at the same time. The strongest lower bounds on competitive ratio for oblivious and adaptive adversary differ. And as both are tight for 2-bounded instances, it seems impossible to obtain an upper bound smaller than 4/3 on the competitive ratio of Mix-R for any non-trivial restriction of the problem in the oblivious adversary model.

In both the algorithm and its analysis it is the respective order of packets' deadlines rather than their exact values that matter. Therefore, our results are also applicable to the *Collecting Items* problem [4], briefly described in Section 1.3. As mentioned in Section 1.4, Mix-R is the optimum randomized memoryless algorithm for *Collecting Items*, cf. Appendix A.

Therefore, to beat either the general bound of $e/(e-1)$, or any of the $1/\left(1-(1-\frac{1}{s})^s\right)$ bounds for s-bounded instances for buffer management with bounded delay, one either needs to consider algorithms that are not memoryless scale-invariant, or better utilize the knowledge of exact deadlines—in the analysis at least, if not in the algorithm itself.

Last but not least, let us remark again that Mix-R and its analysis might automatically provide better bounds for further restricted variants of the problem, provided that some insight allows to confine the adversary's choice of packets for transmission in a step, while knowing which packets are pending for it—one such example is the algorithm for similarly ordered instances (aka instances with agreeable deadlines) [15], as we discussed in Section 2.4.

References

[1] N. Andelman, Y. Mansour, and A. Zhu. Competitive queueing policies for qos switches. In *Proc. of the 14th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 761–770, 2003.

[2] Y. Azar. Online packet switching. In *Proc. of the 2nd Workshop on Approximation and Online Algorithms (WAOA)*, pages 1–5, 2004.

[3] S. Ben-David, A. Borodin, R. M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in online algorithms. *Algorithmica*, 11(1):2–14, 1994. Also appeared in *Proc. of the 22nd STOC*, pages 379–386, 1990.

[4] M. Bienkowski, M. Chrobak, C. Dürr, M. Hurand, A. Jeż, Ł. Jeż, and G. Stachowiak. Collecting weighted items from a dynamic queue. In *Proc. of the 20th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1126–1135, 2009.

[5] M. Bienkowski, M. Chrobak, and Ł. Jeż. Randomized algorithms for buffer management with 2-bounded delay. In *Proc. of the 6th Workshop on Approximation and Online Algorithms (WAOA)*, pages 92–104, 2008.

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

[7] 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. *Journal of Discrete Algorithms*, 4:255–276, 2006.

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

[9] M. Chrobak, W. Jawor, J. Sgall, and T. Tichý. Improved online algorithms for buffer management in QoS switches. *ACM Transactions on Algorithms*, 3(4), 2007. Also appeared in *Proc. of the 12th ESA*, pages 204–215, 2004.

[10] M. Englert and M. Westermann. Considering suppressed packets improves buffer management in QoS switches. In *Proc. of the 18th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 209–218, 2007.

[11] L. Epstein and R. van Stee. Buffer management problems. *Sigact News*, 35:58–66, 2004.

[12] M. Goldwasser. A survey of buffer management policies for packet switches. *SIGACT News*, 41(1):100–128, 2010.

[13] 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.

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

[15] Ł. Jeż. Randomized algorithm for agreeable deadlines packet scheduling. In *Proc. of the 27th Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 489–500, 2010.

[16] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. *SIAM Journal on Computing*, 33(3):563–583, 2004. Also appeared in *Proc. of the 33rd STOC*, pages 520–529, 2001.

[17] F. Li, J. Sethuraman, and C. Stein. An optimal online algorithm for packet scheduling with agreeable deadlines. In *Proc. of the 16th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 801–802, 2005.

[18] A. Zhu. Analysis of queueing policies in QoS switches. *Journal of Algorithms*, 53(2):137–168, 2004.

A Lower Bound for *Collecting Items*

In this section, for completeness, we evoke the lower bound on the *Collecting Items* problem [4]. As the proof is omitted in the original article due to space constraints, and the original theorem statement therein is not parametrized by N, we restate the theorem.

Theorem 5 (Theorem 6.3 of [4]). *For every randomized memoryless algorithm for the* Collecting Items *problem, there is an adaptive adversary's strategy using at most N different packet weights such that the algorithm's competitive ratio against the strategy is at least $1/\left(1-(1-\frac{1}{N})^N\right)$, and at every step the algorithm has at most N packets in its queue.*

Below we present the original proof from [4].

Proof. Fix some online memoryless randomized algorithm A, and consider the following scheme. Let $a > 1$ be a constant, which we specify later, and let $n = N-1$ At the beginning, the adversary inserts items $a^0, a^1, \ldots, a^n$ into the queue, in this order. (To simplify notation, in this proof we identify items with their weights.) In our construction we maintain the invariant that in each step, the list of items pending for A is equal to $a^0, a^1, \ldots, a^n$. Since A is memoryless, in each step it uses the same probability distribution $(q_j)_{j=0}^n$, where q_j is the probability of collecting item a^j. Moreover, $\sum_{i=0}^n q_i = 1$, as without loss of generality the algorithm always makes a move.

We consider $n+1$ strategies for an adversary, numbered $0, 1, \ldots, n$. The k-th strategy is as follows: in each step collect a^k, delete items $a^0, a^1, \ldots, a^k$, and then issue new copies of these items. Additionally, if A collected a^j for some $j > k$, then the adversary issues a new copy of a^j as well. This way, in each step exactly one copy of each a^j is pending for A, while the adversary accumulates in its pending set copies of the items a^j, for $j > k$, that were collected by A.

This step is repeated $T \gg n$ times, and after the last step both the adversary and the algorithm collect all their pending items. Since $T \gg n$, we only need to focus on the expected amortized profits (defined below) in a single step.

We look at the gains of A and the adversary in a single step. If the adversary chooses strategy k, then it gains a^k. Additionally, at the end it collects the item collected by the algorithm if this item is greater than a^k. Thus, its *amortized expected gain* in a single step is $a^k + \sum_{i>k} q_i a^i$. The expected gain of A is $\sum_i q_i a^i$.

For any probability distribution $(q_j)_{j=0}^n$ of the algorithm, the adversary chooses a strategy k which maximizes the competitive ratio. Thus, the competitive ratio of A is is at least

$$R \;=\; \max_k \left\{ \frac{a^k + \sum_{j>k} q_j a^j}{\sum_j q_j a^j} \right\} \;\geq\; \sum_k v_k \frac{a^k + \sum_{j>k} q_j a^j}{\sum_j q_j a^j} \quad , \qquad (14)$$

for any coefficients $v_0, \ldots, v_n \geq 0$ such that $\sum_k v_k = 1$. Let $M = a^{n+1} - n(a-1)$. For $k = 0, 1, ..., n$, we choose

$$v_k \;=\; \begin{cases} \frac{1}{M} a^{n-k}(a-1), & \text{if } k < n \ , \\ \frac{1}{M}\,(a \ - n(a-1),) & \text{if } k = n \ . \end{cases}$$

The choice of these values may seem somewhat mysterious, but it's in fact quite simple—it is obtained by considering A's distributions where $q_j = 1$ for some j (and thus when A is deterministic), assuming that the resulting lower bounds on the right-hand side of (14) are equal, and solving the resulting system of equations.

For these values of v_k we obtain

$$\begin{aligned}
MR\sum_{j=0}^{n} q_j a^j &\geq \sum_{k=0}^{n} Mv_k a^k + \sum_{k=0}^{n} Mv_k \sum_{j>k} q_j a^j \\
&= \sum_{k=0}^{n-1} Mv_k a^k + Mv_n a^n + \sum_{j=0}^{n} q_j a^j \sum_{k<j} Mv_k \\
&= n(a-1)a^n + [a - n(a-1)]a^n + \sum_{j=0}^{n} q_j (a^j - 1)a^{n+1} \\
&= a^{n+1} + a^{n+1}\sum_{j=0}^{n} q_j a^j - a^{n+1}\sum_{j=0}^{n} q_j \\
&= a^{n+1} + a^{n+1}\sum_{j=0}^{n} q_j a^j - a^{n+1} \\
&= a^{n+1}\sum_{j=0}^{n} q_j a^j \quad .
\end{aligned}$$

Therefore, $R \geq a^{n+1}/M$. This bound is maximized for $a = 1 + 1/n$, in which case we get

$$R \geq \frac{\left(1+\frac{1}{n}\right)^{n+1}}{\left(1+\frac{1}{n}\right)^{n+1} - 1} = \frac{\left(1+\frac{1}{N-1}\right)^{N}}{\left(1+\frac{1}{N-1}\right)^{N} - 1} = \frac{1}{1-\left(1-\frac{1}{N}\right)^{N}} \quad ,$$

completing the proof. $\square$