Randomized Two-Valued Bounded Delay Online Buffer Management

Christoph Dürr, Shahin Kamali

Randomized Two-Valued Bounded Delay Online Buffer Management

Christoph Dürr* Shahin Kamali†

November 18, 2020

Abstract

In the bounded delay buffer management problem unit size packets arrive online to be sent over a network link. The objective is to maximize the total weight of packets sent before their deadline. In this paper we are interested in the two-valued variant of the problem, where every packet has either low (1) or high priority weight ($\alpha > 1$). We show that the optimal randomized competitive ratio against an oblivious adversary is $1 + (\alpha - 1)/(\alpha^2 + \alpha)$.

Keywords: competitive ratio, oblivious adversary, buffer management, maximum throughput scheduling.

1 Introduction

Online Buffer Management is a scheduling problem which arises in network routers. Packets arrive online to be sent on a specific link, the goal is to maximize the total weight of sent packets under various constraints. Different models have been studied in the past, the FIFO model, where the router stores pending packets in a limited capacity buffer, and the bounded delay model where packets can stay only limited time in the buffer.

We are particularly interested in the latter model, which can be formalized as the following scheduling problem. Time is partitioned into *time slots*. At each time slot, a (potentially empty) set of jobs of unit length arrive online on a single machine. Job j arrives at a release time $r_j \in \mathbb{N}$, has a deadline $d_j \in \mathbb{N}$ and a weight $w_j \in \mathbb{R}^+$. At every time slot $t \in \mathbb{N}$, there is a set of *pending jobs*, which is updated by jobs released at time t and by jobs expiring at time t. The algorithm can choose to schedule one of the pending jobs, at every time slot The goal is to maximize the total weight of the scheduled jobs. The usual restriction on the online algorithm is that these decisions need to be made without knowledge of future arriving jobs. Its performance is measured by the competitive ratio, which compares the objective value reached by the algorithm with the objective value of the optimal schedule, maximizing the ratio over all possible instances. The deterministic competitive ratio of the problem is defined as the best competitive ratio among all deterministic online algorithms. As randomization usually helps in these online settings, we consider

*Sorbonne Université, CNRS, Laboratoire d'informatique de Paris 6, LIP6, Paris, France. Corresponding author: christoph.durr@lip6.fr. https://orcid.org/0000-0001-8103-5333

†Department of Computer Science, University of Manitoba, Winnipeg, Canada. shahin.kamali@umanitoba.ca. https://orcid.org/0000-0003-1404-2212

the randomized competitive ratio against an oblivious adversary. The term *adversary* comes from the game theoretical setting of an online problem, played between an online algorithm which needs to make decisions at every time slot and a malicious adversary which generates the instance over time. The adversary is "oblivious" in the sense that it is not aware of the random choices made by the randomized online algorithm. Other adversarial models exist but are not considered in this paper.

2 Related work

This problem has been introduced in 2001, together with a lower bound of $\varphi \approx 1.618$ on the deterministic competitive ratio [1] and [2] and an upper bound of 2, achieved by the GREEDY algorithm which schedules always the heaviest pending job (ties are broken by executing the most urgent job). This gap between lower and upper bound generated a series of studies of variants of this problem, mostly with restrictions on the deadlines of the job, such as agreeable deadlines (jobs can be ordered by release times and deadlines at the same time) or s-uniform or s-bounded instances ($d_j - r_j$ is s or at most s) for some parameter $s > 1$. Eventually in SODA 2019, the gap was closed, with a sophisticated algorithm and analysis [3]. The randomized competitive ratio in the oblivious adversary model of this problem is still open, and the optimal competitive ratio is known to be between 1.25 [4] and 1.582 [5, 6].

The paper [7] considered a variant of the problem, where every job has not unit processing time, but a processing time $p \in \mathbb{N}$, all jobs are tight in the sense $d_j = r_j + p$, and the algorithm can preempt job executions. In this setting, the authors give a barely random algorithm with competitive ratio 2, and showed that this is optimal for barely random algorithms. By scaling, this is equivalent to the problem with unit processing time jobs by fractional release times and deadlines. In our problem these times are restricted to integers, and in this context preemption is of no help to the algorithm. We refer to [8] for a survey on packet scheduling, as well as to the more recent introduction of [3] and references therein.

In this paper we consider a particular variant of the bounded delay buffer management problem, where every job can have either low or high weight, that is $w_j \in \{1, \alpha\}$ for some parameter $\alpha > 1$. The 2-valued variant has been studied in [9] who showed the deterministic competitive ratio is $\min\{1 + 1/\alpha, 1 + (\alpha - 1)/(\alpha + 1)\}$. It is motivated by the fact that some applications, as for example online games, could send packets with two kind of informations: Major updates in the game configuration (such as the arrival or departure of participants), and minor updates (such as not critical moves).

2.1 Contributions

The lower bound of the deterministic competitive ratio for the 2-valued variant can be easily transformed into a randomized lower bound. In addition this transformation suggests a randomized online algorithm, which we show to have optimal competitive ratio $1 + (\alpha - 1)/(\alpha^2 + \alpha)$.

3 Preliminaries

Formally an instance of the bounded delay buffer management problem consists of a finite sequence of jobs. Each job j has a unit processing time, a release time $r_j \in \mathbb{N}$, a deadline $d_j \in \mathbb{N}$ and a priority

weight $w_j \in \mathbb{R}_+$. The time line consists of time slots $t \in \mathbb{N}$. During each time slot, at most one job can be executed. A job is pending at time t if $r_j \leq t < d_j$ and if it has not already been scheduled. An online algorithm A can schedule at each time slot $t \in \mathbb{N}$ at most one pending job, and has to make this decision without knowledge of future released jobs. The objective value of a schedule is the total priority weight of all scheduled jobs. This value is compared with the objective value of an optimal schedule, and the resulting value is called the competitive ratio. The optimal schedule can be computed offline by solving a maximum weight bipartite matching problem, matching jobs to time slots. The competitive ratio of algorithm A is the supremum of its competitive ratio over all instances. The competitive ratio of the problem is the infimum of the competitive ratio over all online algorithms. Usually the online computation paradigm is seen as a game played between the algorithm (making decisions which job to schedule) and the adversary (generating jobs).

If A is a randomized algorithm, then the performance of A is the expected objective value of the produced schedule. Different adversarial models have been studied in the literature. In this paper we focus on the oblivious adversary, where the input is generated without knowledge of the random choices made by the algorithm. R5cm

Online algorithms for this problem heavily rely on the notion of a *provisional schedule*, which was introduced in [9]. Recall, that for an algorithm A and a point in time t, we define the set of *pending* jobs as the set of jobs released prior or at time t, which have their deadline strictly after t, and which have not yet been scheduled by A. The provisional schedule S is a maximum weight subset of pending jobs, which can be scheduled from time t on. This set S can be computed greedily, by starting with $S = \emptyset$, and processing first all pending heavy jobs, then all pending light jobs, in arbitrary order within each category. Each considered job j is then added to S if and only if the number of jobs $i \in S$ with $d_i \leq d_j$ is strictly less than $d_j - t$. This condition ensures that there is a provisional feasible schedule for S.

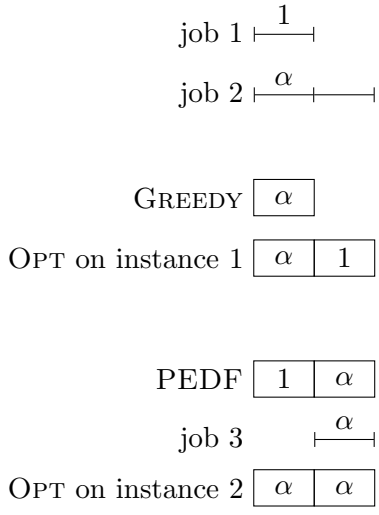

Figure 1: The deterministic lower bound from [9].

There are two natural deterministic algorithms that are defined in [9]. The first algorithm, which we call GREEDY, schedules at every time step, a job of the provisional schedule of maximum weight, breaking ties by the deadlines of job (earliest deadline has the highest priority). The second algorithm, we which we call Provisional Earliest Deadline First (PEDF), schedules at every time step, a job of the provisional schedule of minimum deadline. Note that it differs from the usual EDF algorithm (*earliest deadline first*), which is not restricted to select jobs from the provisional schedule.

The deterministic lower bound consists of an adversary which releases at time 0 two jobs, see Figure 1. A job of weight 1 and deadline 1 and a job of weight α and deadline 2. Any deterministic deterministic algorithm either schedules the urgent job, or the heavy job, or no job at all. Clearly the last option is suboptimal, hence we can focus on the first two options. If the algorithm schedules the heavy job, then no more jobs are released, and the competitive ratio is $\frac{1+\alpha}{\alpha}$. If the algorithm schedules the urgent light job, then at time 1 the adversary releases another job of weight α and deadline 2, leading to the ratio $\frac{2\alpha}{1+\alpha}$. This shows a lower bound of $\min\{\frac{1+\alpha}{\alpha}, \frac{2\alpha}{1+\alpha}\}$ on the deterministic competitive ratio. It also suggests a simple deterministic algorithm. Let $\alpha^* = 1+\sqrt{2}$ be the solution to the equation $\frac{1+\alpha}{\alpha} = \frac{2\alpha}{1+\alpha}$. If $\alpha \leq \alpha^*$ run PEDF, else run GREEDY. In [9] this

strategy was shown to be optimal.

4 Randomized competitive ratio

The previous lower bound can be extended into a lower bound against an oblivious randomized adversary.

Proposition 1 *No randomized online algorithm can achieve a competitive ratio better than* $R = 1 + \frac{\alpha-1}{\alpha^2+\alpha}$.

Proof Let σ_1, σ_2 be the two instances from the previous lower bound construction. Let y be a distribution on σ_1, σ_2. We denote $\mathbb{E}_y[\mathrm{OPT}(\sigma_j)]$ the expected optimal profit, and by $\mathbb{E}_y[\mathrm{A}(\sigma_j)]$ the expected profit of a given randomized algorithm A. By Yao's principle, see [10, Theorem 8.3], the randomized competitive ratio against an oblivious adversary is lower bounded by

$$\max_y \min_A \mathbb{E}_y[\mathrm{OPT}(\sigma_j)]/\mathbb{E}_y[\mathrm{A}(\sigma_j)].$$

We choose $y = (\frac{\alpha-1}{\alpha}, 1/\alpha)$, which leads to the expected optimal profit

$$\mathbb{E}_y[\mathrm{OPT}(\sigma_j)] = \frac{\alpha-1}{\alpha}(1+\alpha) + \frac{1}{\alpha}2\alpha = 2 + \alpha - 1/\alpha.$$

We recall that at time 0, the instances σ_1 and σ_2 are indistinguishable. Any deterministic algorithm has 3 options at time 0, to execute the urgent light job, to execute the heavy job or to remain idle. We ignore the last option, as it is clearly suboptimal. Any deterministic algorithm, which starts by executing the urgent light job, has an expected profit of at most

$$y_1(1+\alpha) + y_2(1+\alpha) = 1 + \alpha.$$

In addition any deterministic algorithm, which starts by executing the heavy job, has an expected profit of at most

$$y_1\alpha + y_2 2\alpha = \frac{\alpha-1}{\alpha}\alpha + \frac{1}{\alpha}2\alpha = 1 + \alpha.$$

This shows the claimed lower bound. □

Note that the above lower bound is maximized at $\alpha = 1+\sqrt{2}$ and gives the value $4-2\sqrt{2} \approx 1.172$.

4.1 Upper bound

The lower bound construction suggests a simple barely random algorithm. At time 0, the algorithm flips a biased coin and based on its outcome decides between the two already mentioned deterministic algorithms. In other words, Barely-Random will run Greedy with probability p and PEDF with probability $1-p$, for

$$p = \frac{\alpha^2-1}{\alpha^2+2\alpha-1}.$$

This surprisingly simple algorithm achieves the optimal competitive ratio.

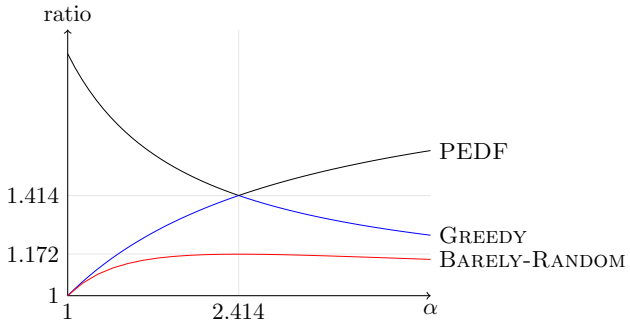

Figure 2: Competitive ratios as function of α. The deterministic competitive ratio is the minimum of the competitive ratios of the algorithms PEDF and Greedy.

Theorem 1 *Algorithm* Barely-Random *has a competitive ratio of* $\frac{\alpha^2+2\alpha-1}{\alpha^2+\alpha}$ *against an oblivious adversary.*

Proof Fix an instance, and denote by Opt an optimal schedule. We start with showing some structural properties of the three schedules Opt, Greedy and PEDF. Consider a time slot, where Greedy executes a heavy job j, while PEDF executes a light job. By the definition of PEDF we known that PEDF will eventually execute j as well.

In addition we observe that Greedy executes a maximum number of heavy jobs while PEDF executes a maximum number of jobs regardless of their weight. This follows by the *earliest deadline policy* applied by both algorithms, respectively on heavy pending jobs or on all pending jobs. Indeed this policy is known to maximize the number of executed jobs for the unweighted unit length job scheduling problem. Therefore OPT does not schedule more jobs than PEDF and does not schedule more heavy jobs than Greedy. Also since PEDF maximizes the number of pending jobs in every time slot, we know that whenever PEDF is idle, then Greedy is idle as well.

As a result, for the analysis of the competitive ratio it suffices to count the number of light and heavy jobs executed by respectively Greedy and PEDF. To formalize this intuition, we use the following notation.

set	content
D	time slots where Greedy is idle but not PEDF
H^*	heavy jobs executed only by Greedy
H	heavy jobs executed both by Greedy and PEDF
L	light jobs executed by Greedy
L'	light jobs executed by PEDF

Note that the jobs executed by PEDF are $H \cup L'$ and the jobs executed by Greedy are $H \cup H^* \cup L$. We use the corresponding lower case notation for the cardinality of those sets. The key to the proof is the following claim.

Claim 1 $d + h^* \leq h$.

The proof uses an injective mapping from L' to $H \cup L$. This implies $\ell' \leq h + \ell$, and by the following

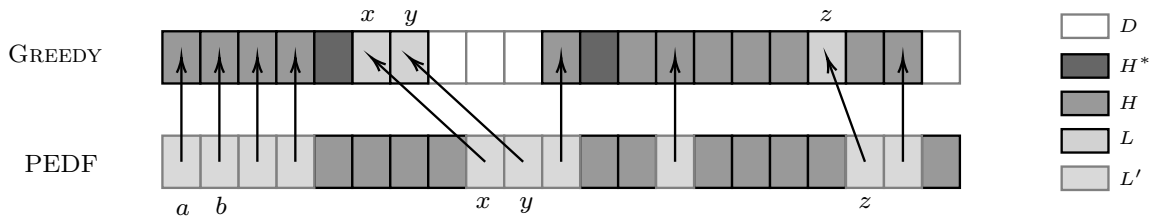

Figure 3: The injective mapping from L' to $H \cup L$. Each job in L' is mapped to either a heavy job in H (e.g., a, b) or it is mapped to itself (e.g., x, y, and z).

equality, the claim holds:

$$d + h^* + h + \ell = h + \ell' \tag{1}$$

The left and right sides of the above equality respectively indicate the number of time slots throughout the execution of GREEDY and PEDF.

To define the mapping, let $j \in L'$ be an arbitrary light job executed by PEDF at some moment t. If GREEDY executes j as well and not later than t, then we map j to itself. Otherwise, we map j to the job k executed by GREEDY at time t. We observe that if k is a heavy job, then it is common to both schedules. This follows by the definition of PEDF, for which k was pending at this point, hence $k \in H$. To show the injective nature of the mapping, for the sake of contradiction, we assume that two jobs $j, k \in L'$ are mapped to the same job $k \in L$, executed by GREEDY at some time t. Then by the definition of the mapping, k must be scheduled after j by PEDF, so both jobs j, k were pending at time t for PEDF but also for GREEDY. This contradicts the fact that both schedules apply the same earliest deadline policy among the light jobs, and concludes the proof of the claim.

To summarize, we can express the objective values of the schedules as

$$\begin{aligned} \text{profit}(\text{OPT}) &\leq h^*\alpha + h\alpha + \ell + d \\ \text{profit}(\text{GREEDY}) &= h^*\alpha + h\alpha + \ell \\ \text{profit}(\text{PEDF}) &= h\alpha + \ell + h^* + d, \end{aligned}$$

where the last expression uses (1).

Claim 1 implies the following inequality, which through a sequence of equivalent transformations transforms into the required bound on the competitive ratio:

$$\begin{aligned} h\alpha + \ell &\geq h^*\alpha + d\alpha \\ h\alpha(\alpha-1) + \ell(\alpha-1) &\geq h^*\alpha(\alpha-1) + d\alpha(\alpha-1) \\ -(h\alpha + h^*\alpha + \ell) + \alpha(h\alpha + h^* + \ell + d) &\geq h^*\alpha(\alpha-1) + d\alpha^2 \\ -(h\alpha + h^*\alpha + \ell) + 2\alpha(h\alpha + h^* + \ell + d) &\geq \alpha(h\alpha + h^* + \ell + d) + h^*\alpha(\alpha-1) + d\alpha^2 \\ \alpha^2(h\alpha + h^*\alpha + \ell) - (h\alpha + h^*\alpha + \ell) + 2\alpha(h\alpha + h^* + \ell + d) &\geq \alpha^2(h\alpha + h^*\alpha + \ell) + \alpha(h\alpha + h^*\alpha + \ell + d) + d\alpha^2 \\ (\alpha^2-1)(h\alpha + h^*\alpha + \ell) + 2\alpha(h\alpha + h^* + \ell + d) &\geq (\alpha^2+\alpha)(h\alpha + h^*\alpha + \ell + d) \\ \frac{\alpha^2-1}{\alpha^2+\alpha}(h\alpha + h^*\alpha + \ell) + \frac{2\alpha}{\alpha^2+\alpha}(h\alpha + h^* + \ell + d) &\geq h\alpha + h^*\alpha + \ell + d \\ R \cdot p \cdot \text{profit}(\text{GREEDY}) + R \cdot (1-p) \cdot \text{profit}(\text{PEDF}) &\geq \text{profit}(\text{OPT}) \\ R \cdot \text{profit}(\text{BARELY-RANDOM}) &\geq \text{profit}(\text{OPT}). \end{aligned} \tag{2}$$

□

5 Conclusion

Now that the randomized competitive ratio of the two valued bounded delay online buffer management problem is tackled — against the oblivious adversary — it would be interesting to see how it can be generalized to more values for the job weights, for example weights belonging to the set $\{1, \alpha, \alpha^2\}$, or even to the general problem. Currently the only known randomized algorithms for the general problem are RMIX and REMIX.

The algorithm RMIX was defined in [5, 6] as follows. At every time slot, consider j the heaviest pending job. Denote w_j its weight. Let $x \in [-1, 0]$ be a uniformly chosen real number. Let f be a job with minimal deadline and weight $w_f \geq e^x w_j$. Execute f in this time slot. We can observe that, if executed on instance 1 of the lower bound instance from Proposition 1, at time 0, it will choose the heavy job with probability $\min\{1, \ln \alpha\}$. Hence it has a larger tendency to choose the heavy job than BARELY-RANDOM, and is fooled by instance 1.

The algorithm REMIX was defined in [11], and behaves even differently on instance 1. It will execute the heavy job with probability $1 - 1/\alpha$ and the light job with probability $1/\alpha$, but is also not optimal. These comparisons with BARELY-RANDOM are unfair in a sense, because they have been designed for the general weighted case, and mostly for an adaptive adversary. But the comparison indicates room for improvement. The optimal randomized algorithm for the general weighted problem, might combine ideas of all these mentioned algorithms, and in particular borrow from the optimal deterministic algorithm [3].

6 Acknowledgment

This work was partially supported by the French research agency (ANR-19-CE48-0016), the CNRS PEPS project ADVICE as well as by the EPSRC grant EP/S033483/1, and by the NSERC Discovery Grant.

We would like to thank Spyros Angelopoulos for helpful discussions as well as anonymous referees for spotting an error in a previous version of this paper.

References

[1] B. Hajek, On the competitiveness of on-line scheduling of unit-length packets with hard deadlines in slotted time, Proc. of the 2001 Conference on Information Sciences and Systems, 2001, pp. 434–439.

[2] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, M. Sviridenko, Buffer Overflow Management in QoS Switches, SIAM Journal on Computing 33 (3) (2004) 563–583.

[3] P. Veselý, M. Chrobak, Ł. Jeż, J. Sgall, A Φ-Competitive Algorithm for Scheduling Packets with Deadlines, Proc. of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2019, pp. 123–142.

[4] F. Y. L. Chin, S. P. Y. Fung, Online Scheduling with Partial Job Values: Does Timesharing or Randomization Help?, Algorithmica 37 (3) (2003) 149–164.

[5] Y. Bartal, F. Y. L. Chin, M. Chrobak, S. P. Y. Fung, W. Jawor, R. Lavi, J. Sgall, T. Tichý, Online Competitive Algorithms for Maximizing Weighted Throughput of Unit Jobs, Proc. of the 21th Symposium on Theoretical Computer Science, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2004, pp. 187–198.

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

[7] F. Y. L. Chin, S. P. Y. Fung, Improved competitive algorithms for online scheduling with partial job values, Theoretical Computer Science 325 (3) (2004) 467–478.

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

[9] M. Englert, M. Westermann, Considering Suppressed Packets Improves Buffer Management in Quality of Service Switches, SIAM Journal on Computing 41 (5) (2012) 1166–1192.

[10] A. Borodin, R. El-Yaniv, Online computation and competitive analysis, Cambridge University Press, 2005.

[11] Ł. Jeż, A Universal Randomized Packet Scheduling Algorithm, Algorithmica 67 (4) (2013) 498–515.