Improved Online Algorithms for Buffer Management in QoS Switches

MAREK CHROBAK and WOJCIECH JAWOR
University of California
JIŘÍ SGALL and TOMÁŠ TICHÝ
Academy of Sciences of the Czech Republic

We consider the following buffer management problem arising in QoS networks: packets with specified weights and deadlines arrive at a network switch and need to be forwarded so that the total weight of forwarded packets is maximized. Packets not forwarded before their deadlines are lost. The main result of the paper is an online $64/33 \approx 1.939$-competitive algorithm – the first deterministic algorithm for this problem with competitive ratio below 2. For the 2-uniform case we give an algorithm with ratio ≈ 1.377 and a matching lower bound.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms: Algorithms

Additional Key Words and Phrases: Online algorithms, scheduling

1. INTRODUCTION

One of the issues arising in IP-based QoS networks is how to manage the packet flows at a router level. In particular, in case of overloading, when the total incoming traffic exceeds the buffer size, the buffer management policy needs to determine which packets should be dropped by the router. Kesselman et al. [2001; 2004] postulate that the packet drop policies can be modeled as combinatorial optimization problems. Of the two models proposed in [Kesselman et al. 2001; 2004], the one relevant to this work is called *buffer management with bounded delay*, and is defined as follows: Packets arrive at a network switch. Each packet is characterized by a positive weight and a deadline before which it must be transmitted. Packets can only be transmitted at integer time steps, one packet at a time. If the deadline of

Author's addresses: Marek Chrobak and Wojciech Jawor, Department of Computer Science, University of California, Riverside, CA 92521. {marek,wojtek}@cs.ucr.edu.
Jiří Sgall and Tomáš Tichý, Mathematical Institute, Academy of Sciences of the Czech Republic, Žitná 25, CZ-11567 Praha 1, Czech Republic. {sgall,tichy}@math.cas.cz.
M. Chrobak and W. Jawor were supported by NSF grants CCR-9988360, CCR-0208856, and CCF0729071. J. Sgall and T. Tichý were partially supported by Institutional Research Plan No. AV0Z10190503, by Inst. for Theor. Comp. Sci., Prague (project 1M0545 of MŠMT ČR), grant 201/05/0124 of GA ČR, and grant IAA1019401 of GA AV ČR.

a packet is reached while it is still being buffered, the packet is lost. The goal is to maximize the total weight of the forwarded packets.

This buffer management problem is equivalent to the online version of the following single-machine unit-job scheduling problem. We are given a set of unit-length jobs, with each job j specified by a triple (r_j, d_j, w_j), where r_j and d_j are integral release times and deadlines, and w_j is a non-negative real weight. One job can be processed at each integer time. We use the term *weighted throughput* or *gain* for the total weight of the jobs completed by their deadline. The goal is to compute a schedule that maximizes the weighted throughput.

In the online version of the problem jobs arrive at their release times. At each time step the algorithm needs to choose and schedule one of the pending jobs, without knowing the jobs released later in the future. An online algorithm $\mathcal{A}$ is *R-competitive* if its gain on any instance is at least $1/R$ times the optimal (offline) gain on this instance. The *competitive ratio* of $\mathcal{A}$ is the infimum of such values R. It is standard to view the online problem as a game between an online algorithm $\mathcal{A}$ and an *adversary*, who issues the jobs and schedules them in order to maximize the ratio between his gain and the gain of $\mathcal{A}$.

We remark that in the literature on online algorithms (see, e.g., [Borodin and El-Yaniv 1998]), an additive constant is often allowed in the definition of competitiveness. More specifically, in order to be R-competitive, $\mathcal{A}$'s gain on any instance needs to be at least $1/R$ times the optimal gain, plus B, where B is a constant independent of the instance. Since the problem allows arbitrary weights, it is scalable and each algorithm can be modified so that the constant B is arbitrarily small. In the algorithms for unit job scheduling presented in our paper, as well as those in the literature, the constant B is in fact zero. Further, all the known lower bounds on the competitive ratio discussed later in the introduction apply to this more general definition with an additive constant.

Past work. A simple greedy algorithm that always schedules the heaviest available job is 2-competitive, and this is the best previous bound for deterministic algorithms for this problem. A lower bound of $\phi \approx 1.618$ was shown in [Andelman et al. 2003; Chin and Fung 2003; Hajek 2001].

Some restrictions on instances of the problem have been studied in the literature [Kesselman et al. 2001; 2004; Andelman et al. 2003; Chin and Fung 2003; Li et al. 2005]. Let the *span* of a job be the difference between its deadline and the release time. In *s-uniform instances* the span of each job is equal exactly s. In *s-bounded* instances, the span of each job is at most s. The lower bound of $\phi \approx 1.618$ in [Andelman et al. 2003; Chin and Fung 2003; Hajek 2001] applies even to 2-bounded instances. A matching upper bound for the 2-bounded case was presented in [Kesselman et al. 2001; 2004]. Algorithms for 2-uniform instances were studied by Andelman et al. [2003], who established a lower bound of $\frac{1}{2}(\sqrt{3}+1) \approx 1.366$ and an upper bound of $\sqrt{2} \approx 1.414$. This upper bound is tight for memoryless algorithms [Bartal et al. 2004; Chin et al. 2006], that is, algorithms which base their decisions only on the weights of pending jobs and are invariant under scaling of weights. Finally, the first deterministic algorithms with competitive ratio lower than 2 for the s-bounded instances appear in [Bartal et al. 2004; Chin et al. 2006]. These ratios, however, depend on s, and approach 2 as s increases. Kesselman et al.

[2003; 2005] gave an algorithm for s-uniform instances with ratio 1.983, independent of s.

Kesselman et al. [2001; 2004; 2003; 2005] consider a different model related to the s-uniform case: packets do not have individual deadlines, but instead they are stored in a FIFO buffer of capacity s (i.e., if a packet is served, all packets in the buffer that arrived before the served one are dropped). Any algorithm in this FIFO model applies also to s-uniform model and has the same competitive ratio [Kesselman et al. 2003; 2005]. Bansal et al. [2004] gave a deterministic 1.75-competitive algorithm in the FIFO model; this implies a 1.75-competitive algorithm for the s-uniform case.

We say that jobs are *similarly ordered* if $r_i < r_j$ implies $d_i \leq d_j$ for all jobs i, j. Note that this includes s-uniform and 2-bounded instances, but not s-bounded ones for $s \geq 3$. Recently, Li et al. [2005] gave a ϕ-competitive algorithm for instances with similarly ordered jobs; this matches the lower bound for 2-bounded instances, and thus it is an optimal algorithm for this case. (Li et al. [Li et al. 2005] use a different terminology for the same restriction on inputs and say that the jobs have *agreeable deadlines*.)

Randomized algorithms have been studied in the literature as well. A randomized 1.582-competitive algorithm for the general case was given in [Bartal et al. 2004; Chin et al. 2006]. For 2-bounded instances, there is a 1.25-competitive algorithm [Bartal et al. 2004; Chin et al. 2006] that matches the lower bound [Chin and Fung 2003]. For the 2-uniform case, the currently best lower bound for randomized algorithms is 1.172 [Bartal et al. 2004].

Our results. We present two deterministic online algorithms for the buffer management problem. Our main result is a deterministic $64/33 \approx 1.939$-competitive algorithm for the general case. This is the first deterministic algorithm for this problem with competitive ratio strictly below 2. All of the algorithms given previously in [Kesselman et al. 2001; 2004; Andelman et al. 2003; Bartal et al. 2004; Chin et al. 2006; Kesselman et al. 2003; 2005] achieved competitive ratio below 2 only when additional restrictions on instances were placed.

With a minor modification, our algorithm yields a $(5 - \sqrt{10})$-competitive algorithm for the case of similarly ordered jobs (where $5 - \sqrt{10} \approx 1.838$). This result appeared in the conference version of this paper [Chrobak et al. 2004], but it was subsequently improved in [Li et al. 2005], and we do not include it in this paper.

We also completely solve the 2-uniform case: we give an algorithm with competitive ratio ≈ 1.377 and a matching lower bound. This ratio is strictly in-between the previous lower and upper bounds from [Andelman et al. 2003]. Our algorithm is rather technical, which is not really surprising, given the lower bound of $\sqrt{2}$ for memoryless algorithms [Bartal et al. 2004; Chin et al. 2006]. To our knowledge, the lower bound of ≈ 1.377 is the best lower bound for the s-uniform case for any s.

Subsequent work. Subsequent to our work, two different and better algorithms for the general case have been obtained by Englert and Westerman [2007] and Li et al. [2007]. The currently best algorithm is 1.828-competitive [Englert and Westerman 2007]. Englert and Westerman [2007] also give a 1.893-competitive memoryless algorithm, solving our previous open problem.

2. TERMINOLOGY AND NOTATION

Unit job scheduling. We present our results in terms of unit job scheduling, as explained in the introduction. A *schedule* S specifies which jobs are executed, and for each executed job j it specifies an integral time t, $r_j \leq t < d_j$, when it is scheduled. (When we say that j is scheduled at time t, we mean that j is started at time t, and thus it occupies the processor through the time interval $[t, t+1)$.) Only one job can be scheduled at any time t. The *throughput* or *gain* of a schedule S is the total weight of the jobs executed in S. If $\mathcal{A}$ is a scheduling algorithm, by $gain_{\mathcal{A}}(I)$ we denote the gain of the schedule computed by $\mathcal{A}$ on an instance I. A job i is *pending* in S at time t if $r_i \leq t < d_i$ and i has not been scheduled in S before t. (Thus all jobs released at time t are considered pending.) An instance is *s-bounded* if $d_j - r_j \leq s$ for all jobs j. Similarly, an instance is *s-uniform* if $d_j - r_j = s$ for all j. The difference $d_j - r_j$ is called the *span* of a job j. An instance is *similarly ordered* if the release times and deadlines are similarly ordered, that is $r_i < r_j$ implies $d_i \leq d_j$ for any two jobs i and j.

Given two jobs i, j, we say that i *dominates* j if either (i) $d_i < d_j$, or (ii) $d_i = d_j$ and $w_i > w_j$, or (iii) $d_i = d_j$, $w_i = w_j$ and $i < j$. (Condition (iii) only ensures that ties are broken in some arbitrary but consistent way.) Given a non-empty set of jobs J, the *dominant* job in J is the one that dominates all other jobs in J; it is always uniquely defined as 'dominates' is a linear order.

A schedule S is called *canonical earliest-deadline* if for any jobs i and j in S, where i is scheduled at time t and j is scheduled later, either j is released strictly after time t, or i dominates j. In other words, at any time, the job to be scheduled dominates all pending jobs that appear later in S. Any schedule can be easily converted into a canonical earliest-deadline schedule by rearranging its jobs. Thus we may assume that offline schedules are canonical earliest-deadline.

3. A $64/33$-COMPETITIVE ALGORITHM

We start with some intuitions that should be helpful in understanding the algorithm and its analysis. The greedy algorithm that always executes the heaviest job is not better than 2-competitive. An alternative idea is to execute the earliest deadline job at each step. This algorithm is not competitive at all, as it could execute many small weight jobs even if there are heavy jobs pending with only slightly larger deadlines. A natural refinement of this approach is to focus on sufficiently heavy jobs, say the jobs of weight at least α times the maximal weight of a pending job, and chose the dominating job among those. As it turns out, this algorithm is also not better than 2-competitive.

The general idea of our new algorithm is to alternatively choose either the heaviest job, or the dominating job with sufficiently large weight (at least α times the maximum weight of a pending job.) Although this simple algorithm, as stated, still has ratio no better than 2, we can reduce the ratio by introducing some minor modifications.

Algorithm GenFlag**:** We use parameters $\alpha = 7/11$, $\beta = 8/11$, and a Boolean variable E, initially set to 0, that stores information about the previous step. At a given time step t, update the set of pending jobs (remove jobs with deadline t and add jobs released at t). If there are no pending jobs, go to the next time step.

Otherwise, let h be the heaviest pending job (breaking ties in favor of dominant jobs) and e the dominant job among the pending jobs with weight at least αw_h. Schedule either e or h according to the following procedure:

```
      if E = 0 then
          if e = h then
(O)           schedule e
          else
              E ← 1
(E)           schedule e
      else
          E ← 0
          if d_e = t + 1 and w_e ≥ βw_h then
(U)           schedule e
          else
(H)           schedule h
```

As indicated in the pseudo-code above, we have four types of jobs. A job e scheduled while $E = 0$ is called an *O-job* if $e = h$, otherwise it is called an *E-job*. A job e scheduled while $E = 1$ is called an *U-job*. A job h scheduled in the last case is called an *H-job*. (The letters stand for **O**bvious, **E**arly, **U**rgent, and **H**eaviest.)

Variable E is 1 iff the previous job was an E-job. Thus, in terms of the labels, the algorithm proceeds as follows: If an O-job is available, we execute it. Otherwise, we execute an E-job, and in the next step either a U-job (if available) or an H-job. (There always is a pending job at this next step: if $d_h = t$ then $e = h$ by the definition of dominance; so if an E-job is scheduled at time t, then $d_h > t$ and h is pending in the next step.)

THEOREM 3.1. GENFLAG *is a* 64/33*-competitive deterministic algorithm for unit-job scheduling.*

PROOF. The proof is by a charging scheme. Fix an arbitrary (offline) schedule ADV. For each job j executed in ADV, we charge its weight w_j to one or two jobs executed by GENFLAG. If the total charge to each job i of GENFLAG were at most Rw_i, the R-competitiveness of GENFLAG would follow by summation over all jobs. Our charging scheme does not always meet this simple condition. Instead, we divide the jobs of GENFLAG into disjoint groups, where each group is either a single O-job, or an EH-pair (an E-job followed by an H-job), or an EU-pair (an E-job followed by a U-job). This is possible by the discussion of types of jobs before the theorem. For each group we prove that its *charging ratio* is at most R, where the charging ratio is defined as the total charge to this group divided by the total weight of the group. This implies that GENFLAG is R-competitive by summation over all groups.

Charging scheme. Let j be the job executed at time t in ADV. Denote by i and h, respectively, the job executed by GENFLAG and the heaviest pending job at time t. (If there are no pending jobs, introduce a "dummy" job of weight 0. This does not change the algorithm.) Then j is charged to GENFLAG's jobs, according to the following rules (see Figure 1).

(EB) If j is executed by GenFlag before time t, then charge $(1-\beta)w_h$ to i and the remaining $w_j - (1-\beta)w_h$ to j. (The latter charge could be negative.)

(EF) Else, if j is executed by GenFlag after time t, then charge βw_j to i and $(1-\beta)w_j$ to j.

(MF) Else, if i is an H-job, $w_j \geq \beta w_i$, and ADV executes i after time t, then charge βw_j to i and $(1-\beta)w_j$ to the job executed by GenFlag at time $t+1$. (In this case, we have $j \neq i$, and j is not executed by GenFlag.)

(AN) Else, charge w_j to i. (Note that this case includes the case $i = j$.)

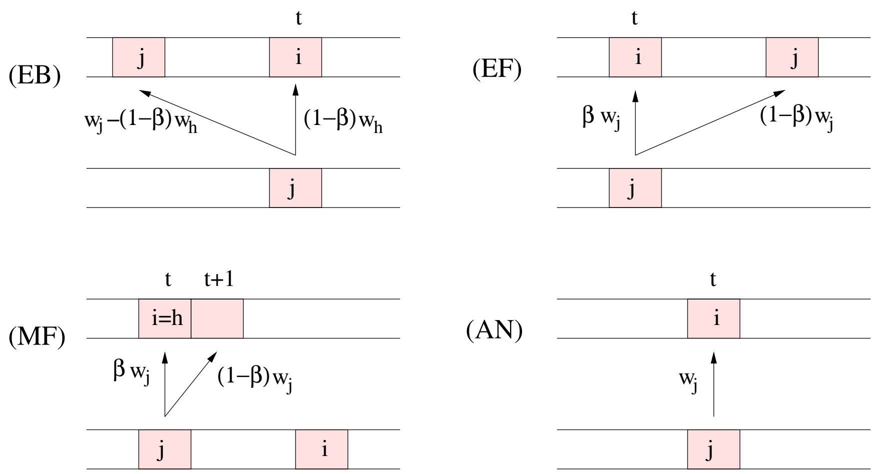

Fig. 1. Illustration of the charging scheme.

We label all charges as EB, EF, MF, AN, according to which case above applies. We also distinguish upward, forward, and backward charges, defined in the obvious way. Thus, for example, in case (EB), the charge of $w_j - (1-\beta)w_h$ to j is a backward EB-charge. (The letters in the labels refer to whether j was executed or missed by GenFlag, and whether j generated backward or forward charge: **E**xecuted-**B**ackward, **E**xecuted-**F**orward, **M**issed-**F**orward, **A**ny-**N**one.)

Each job can receive several charges, but it can get at most one upward charge (from one of the four cases above), and at most one of each remaining type of charges: a backward EB-charge, a forward EF-charge, and a forward MF-charge.

Fix some time t, and let j and i be the jobs scheduled at time t in ADV and in GenFlag, respectively. As in the algorithm, among the jobs pending in GenFlag, let h the heaviest job and e the dominant job with $w_e \geq \alpha w_h$. Of course, $i \in \{e, h\}$. We start with several simple observations used later in the analysis, sometimes without explicit reference. In the following, whenever we say "a pending job" we mean a job which is pending for GenFlag, unless noted otherwise.

Fact 3.2. *If k is a job pending at time t then* (a) $w_k \leq w_h$. *Further,* (b) *if k dominates e then $w_k < \alpha w_h$.*

PROOF. Inequality (a) is trivial, by the definition of h. Part (b) follows from the fact that e dominates all pending jobs with weight at least αw_h. □

FACT 3.3. *Suppose that i receives a forward MF-charge from the job l scheduled at time $t-1$ in* ADV. *Then* (a) *l is not executed by* GENFLAG, (b) $d_l \geq t+1$ *(thus l is pending in* GENFLAG *at time t), and* (c) *this forward MF-charge is at most* $(1-\beta)w_h$.

PROOF. Denote by f the heaviest pending job of GENFLAG at time $t-1$. By the definition of MF-charges, l is not executed by GENFLAG (justyfing (a)), $l \neq f$, f is executed at time $t-1$ as an H-job, $d_f \geq t+1$, and $w_l \geq \beta w_f$. Therefore $d_l \geq t+1$, since otherwise GENFLAG would execute l (or some other job) as a U-job at step $t-1$, proving (b). Part (c) follows from Fact 3.2(a) and the definition of MF-charges. □

FACT 3.4. *Suppose that $i = e$. Then* (a) *the upward charge to e is at most w_h. Further,* (b) *if* ADV *executes e after time t then the upward charge is at most αw_h.*

PROOF. If j is scheduled by GENFLAG before time t, then the upward charge is $(1-\beta)w_h \leq \alpha w_h$ and both parts (a) and (b) hold.
So we can assume now that j is pending at time t. In that case, the upward charge is at most $w_j \leq w_h$, completing the proof of (a). To show (b), since ADV is canonical and it executes e after j, job j must dominate e. By Fact 3.2(b), the upward charge is at most $w_j \leq \alpha w_h$, as claimed. □

FACT 3.5. *Suppose that $i = h$ is executed after time t in* ADV. *Then the upward charge to h is at most βw_h.*

PROOF. In case (EB), the upward charge to h is at most $(1-\beta)w_h \leq \beta w_h$. In all other cases, j is pending at time t, so $w_j \leq w_h$. In cases (EF) and (MF), the charge is $\beta w_j \leq \beta w_h$. In case (AN) the charge is w_j, but since (MF) did not apply, this case can occur only if $w_j \leq \beta w_h$. □

LEMMA 3.6. *Suppose that $i = e$. Then e is charged at most $\alpha w_h + (2-\beta)w_e$.*

PROOF. To prove the lemma, we distinguish several cases.
<u>Case 1</u>: e gets no backward EB-charge. Then, in the worst case, e gets an upward charge of w_h, a forward MF-charge of $(1-\beta)w_h$, and a forward EF-charge of $(1-\beta)w_e$. Using $(2-\beta)w_h = 2\alpha w_h$ and $\alpha w_h \leq w_e$, the total charge is at most $(2-\beta)w_h + (1-\beta)w_e \leq \alpha w_h + (2-\beta)w_e$ as claimed.
<u>Case 2</u>: e gets a backward EB-charge. Then there is no forward EF-charge and the upward charge is at most αw_h by Fact 3.4(b). The backward EB-charge is at most w_e, and thus, if there is no MF-charge, the total charge is at most $\alpha w_h + w_e \leq \alpha w_h + (2-\beta)w_e$, as claimed.
So we can assume that e receives an MF-charge from the job l job scheduled at time $t-1$ in ADV. By Fact 3.3(a), l is not executed by GENFLAG and in particular it is pending for GENFLAG at time t. We have two sub-cases.

<u>Case 2.1</u>: $d_l \geq d_e$. Then l is pending for GENFLAG also at the time t' when ADV schedules e. Consequently, denoting by h' the heaviest pending job at time t', we obtain that e receives at most a backward EB-charge $w_e - (1 -$

$\beta)w_{h'} \le w_e - (1-\beta)w_l$, a forward MF-charge $(1-\beta)w_l$, and an upward charge αw_h. The total is at most $\alpha w_h + w_e \le \alpha w_h + (2-\beta)w_e$, as claimed.

Case 2.2: $d_l < d_e$. In this case, l is pending at t, and l dominates e, so we must have $w_l < \alpha w_h \le w_e$, by Fact 3.2(b). So the forward MF-charge is at most $(1-\beta)w_l \le (1-\beta)w_e$. With the backward EB-charge of at most w_e and upward charge of at most αw_h, the total is at most $\alpha w_h + (2-\beta)w_e$, as claimed.

We have now examined all cases, and the proof of the lemma is complete. □

This completes the proofs of all observations. Now we examine the charges to all job groups in GenFlag's schedule: single O-jobs, EH-pairs, and EU-pairs.

O-jobs. Let $e = h$ be an O-job executed at time t. The forward MF-charge is at most $(1-\beta)w_e$. We have two cases.

Suppose e gets a backward EB-charge. Then e gets no forward EF-charge and the upward charge is at most αw_e; thus the total charging ratio is at most $2+\alpha-\beta < R$.

In the other case, e does not get a backward EB-charge. Then the forward EF-charge is at most $(1-\beta)w_e$ and the upward charge is at most w_e, so the charging ratio is at most $3 - 2\beta < R$.

EH-pairs. Let e be the E-job scheduled at time t, h the heaviest pending job at time t, and h' the H-job at time $t+1$. By the algorithm, $e \ne h$ and $d_h > t+1$ (since GenFlag did not execute h as an O-job at time t.) Thus h is still pending after the E-step and $w_{h'} \ge w_h$.

We now estimate the charge to h'. There is no forward MF-charge, as the previous step is not an H-step. If there is a backward EB-charge, the additional upward charge is at most $\beta w_{h'}$ by Fact 3.5 and the total is at most $(1+\beta)w_{h'}$. If there is no EB-charge, the sum of the upward charge and a forward EF-charge is at most $w_{h'} + (1-\beta)w_{h'} \le (1+\beta)w_{h'}$.

The charge to e is at most $\alpha w_h + (2-\beta)w_e$ by Lemma 3.6. So the total charge of the EH-pair is at most $\alpha w_h + (2-\beta)w_e + (1+\beta)w_{h'}$, and the charging ratio is at most

$$\begin{aligned} 2-\beta+\frac{\alpha w_h + (2\beta-1)w_{h'}}{w_e + w_{h'}} &\le 2-\beta+\frac{\alpha w_h + (2\beta-1)w_{h'}}{\alpha w_h + w_{h'}} \\ &\le 2-\beta+\frac{\alpha+2\beta-1}{\alpha+1} \\ &= R. \end{aligned}$$

The first step follows from $w_e \ge \alpha w_h$. As $2\beta - 1 < 1$, the next expression is decreasing in $w_{h'}$, so the maximum is at $w_{h'} = w_h$ and it is equal to R, by the definitions of α and β.

EU-pairs. As in the previous case, let e and h denote the E-job scheduled at time t and the heaviest pending job at time t. By g and h' we denote the scheduled U-job and the heaviest pending job at time $t+1$. As in the case of EH-pairs, $e \ne h$ and $w_{h'} \ge w_h$.

Job g gets no backward EB-charge, since it expires, and no forward MF-charge, since the previous step is not an H-step. The upward charge is at most $w_{h'}$, the forward EF-charge is at most $(1-\beta)w_g$.

The charge to e is at most $\alpha w_h + (2-\beta)w_e$ by Lemma 3.6. Thus the total charge of the EU-pair is at most $\alpha w_h + (2-\beta)w_e + w_{h'} + (1-\beta)w_g$, and the charging ratio is at most

$$\begin{aligned} 2-\beta+\frac{\alpha w_h + w_{h'} - w_g}{w_e + w_g} &\le 2-\beta+\frac{\alpha w_h + (1-\beta)w_{h'}}{\alpha w_h + \beta w_{h'}} \\ &\le 2-\beta+\frac{\alpha+1-\beta}{\alpha+\beta} \\ &= R. \end{aligned}$$

In the first step, we apply bounds $w_e \ge \alpha w_h$ and $w_g \ge \beta w_{h'}$. As $1-\beta < \beta$, the next expression is decreasing in $w_{h'}$, so the maximum is at $w_{h'} = w_h$.

Summarizing, we now have proved that the charging ratio to all job groups is at most R, and the R-competitiveness of GenFlag follows.

Similarly ordered jobs

As shown in the conference version of this paper [Chrobak et al. 2004], Algorithm GenFlag can be modified to obtain the ratio of $5-\sqrt{10} \approx 1.838$ for similarly ordered instances. In this modified algorithm we use a constant $\alpha = \sqrt{10}/5 \approx 0.633$. At each step we schedule either h or e (these jobs are defined as before, except that the new value of α is used to choose e) according to the following procedure:

```
if E = 0 then
    schedule e
    if e ≠ h ∧ d_e > t + 1 then E ← 1
else
    schedule h
    E ← 0
```

The analysis of this algorithm can be found in [Chrobak et al. 2004].

4. 2-UNIFORM INSTANCES

In this section we consider 2-uniform instances, where each job j satisfies $d_j = r_j + 2$. Let $Q \approx 1.377$ be the largest root of $Q^3 + Q^2 - 4Q + 1 = 0$. First, we prove that no online algorithm for this problem can be better than Q-competitive. Next, we show that this lower bound is in fact tight.

4.1 Lower Bound

The proof is by constructing an appropriate adversary strategy. Given an online algorithm $\mathcal{A}$, the adversary releases a sequence of jobs on which the gain of $\mathcal{A}$ is less than Q times the optimal gain.

At each step t, we distinguish *old pending jobs*, that is, those that were released at time $t-1$ but not executed, from the newly released jobs. We can always ignore all the old pending jobs except for the heaviest one, as only one of the old pending jobs can be executed. To simplify notation, we identify jobs by their weight. Thus "job x" means the job with weight x. Such a job is usually uniquely defined by the context, possibly after specifying if it is an old pending job or a newly released job.

For simplicity, we assume first that the additive constant in the definition of competitiveness is 0. We show later how this assumption can be eliminated.

Fix some $0 < \epsilon < 2Q - 2$. We define a sequence Ψ_i, $i = 1, 2, \ldots$, as follows. For $i = 1$, $\Psi_1 = Q - 1 - \epsilon$. Inductively, for $i \geq 1$, let

$$\Psi_{i+1} \;=\; \frac{(2-Q)\Psi_i - (Q-1)^2}{2 - Q - \Psi_i}.$$

LEMMA 4.1. *For all i, we have $|\Psi_i| < Q - 1$. Furthermore, the sequence $\{\Psi_i\}$ converges to $1 - Q$.*

PROOF. Substituting $z_i = \Psi_i + Q - 1$, we get a recurrence $z_{i+1} = \frac{(3-2Q)z_i}{1-z_i}$. Note that $z_1 = 2Q - 2 - \epsilon$ and that $0 < z_i \leq 2Q - 2 - \epsilon$ implies

$$0 \;<\; z_{i+1} \;\leq\; \frac{3-2Q}{3-2Q+\epsilon} z_i \;<\; z_i.$$

Thus, by induction, $0 < z_i \leq 2Q - 2 - \epsilon$ for all i and, furthermore, $\lim_{i\to\infty} z_i = 0$. The lemma follows immediately. □

THEOREM 4.2. *There is no deterministic online algorithm for the 2-uniform case with competitive ratio smaller than Q.*

PROOF. Let $\mathcal{A}$ be some online algorithm for the 2-uniform case. We develop an adversary strategy that forces $\mathcal{A}$'s ratio to be bigger than $Q - \epsilon$.

Let Ψ_i be as defined before Lemma 4.1. For $i \geq 1$ define

$$a_i \;=\; \frac{1 - \Psi_i}{Q - 1} \quad \text{and} \quad b_i \;=\; \frac{Q(2 - Q - \Psi_i)}{(Q-1)^2}.$$

By Lemma 4.1, for all i, $b_i > a_i > 1$ and, for large i, $a_i \approx 3.653$ and $b_i \approx 9.688$.

Our strategy proceeds in stages. It guarantees that at the beginning of stage $i = 1, 2, \ldots$, both $\mathcal{A}$ and the adversary have one or two old pending job(s) of the same weight x_i. Note that it is irrelevant whether one or two old pending jobs are present, and also whether they are the same for $\mathcal{A}$ and the adversary.

Each stage $i \geq 1$ except last consists of three time steps. The last stage can consist of one, two, or three steps. We will also have an initial stage numbered 0 that consists of one time step.

Initially, in stage 0, we issue two jobs of some arbitrary weight $x_1 > 0$ at time 0. Both $\mathcal{A}$ and the adversary execute one job x_1, and at the beginning of stage 1 both have an old pending job with weight x_1.

At the beginning of stage $i \geq 1$, $\mathcal{A}$ and the adversary start with an old pending job x_i. The adversary now follows this procedure:

```
        issue one job a_i x_i
(A)     if A executes a_i x_i then execute x_i, a_i x_i and halt
        else  (A executes x_i)
            at the next time step issue b_i x_i
(B)         if A executes b_i x_i then execute x_i, a_i x_i, b_i x_i, and halt
            else  (A executes a_i x_i)
(C)             at the next time step issue two jobs x_{i+1} = b_i x_i
                execute a_i x_i, b_i x_i, b_i x_i
```

If $\mathcal{A}$ executes first x_i and then $a_i x_i$, then after step (C) it executes one job $b_i x_i$, either the old pending one or one of the two newly released jobs. After this, both $\mathcal{A}$ and the adversary have one or two newly released jobs $b_i x_i$ pending, and the new stage starts with $x_{i+1} = b_i x_i$.

A single complete stage of the adversary strategy is illustrated in Figure 2.

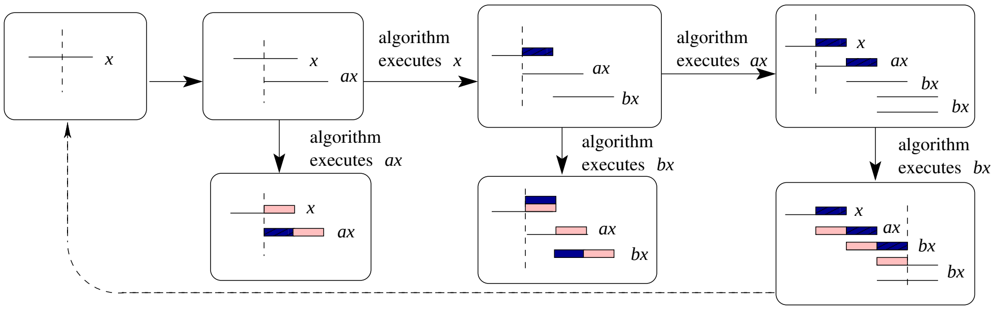

Fig. 2. The adversary strategy. We denote $x = x_i$, $a = a_i$, and $b = b_i$. Line segments represent jobs, dark rectangles represent slots when the job is executed by $\mathcal{A}$, and lightly shaded rectangles represent executions by the adversary.

If the game reaches stage i, then define $gain_i$ and adv_i to be the total gain of $\mathcal{A}$ and the adversary, respectively, in stages $0, 1, \ldots, i-1$. By ρ we denote the sequence of all jobs released by the adversary.

Claim A: For any i, either the game stops before stage i and the algorithm fails to be $(Q-\epsilon)$-competitive on the input sequence ρ, i.e., $(Q-\epsilon)gain_{\mathcal{A}}(\rho) - adv(\rho) < 0$, or else at the beginning of stage i we have

$$(Q-\epsilon)gain_i - adv_i \;\le\; \Psi_i x_i. \tag{1}$$

The proof of Claim A is by induction on the number of stages. For $i = 1$, $(Q-\epsilon)gain_1 - adv_1 \le (Q-\epsilon-1)x_1 = \Psi_1 x_1$, and the claim holds.

In the inductive step, suppose that stage i has been reached and is about to start. Thus now $\mathcal{A}$ and the adversary have an old pending job with weight x_i and (1) holds. If $\mathcal{A}$ executes $a_i x_i$ in step (A), then, denoting by ρ the sequence of all released jobs (up to and including $a_i x_i$), using the inductive assumption, and substituting the formula for a_i, we have

$$\begin{aligned}(Q-\epsilon)gain_{\mathcal{A}}(\rho) - adv(\rho) \;&=\; (Q-\epsilon)gain_i - adv_i + (Q-\epsilon)a_i x_i - (x_i + a_i x_i)\\ &\le\; [\Psi_i - 1 + (Q-\epsilon-1)a_i]x_i \;=\; -\epsilon a_i x_i \;<\; 0,\end{aligned}$$

as claimed.

If $\mathcal{A}$ executes x_i and then $b_i x_i$ in (B), then, again, denoting by ρ the sequence of all released jobs, using the inductive assumption, and substituting the formulas for a_i, b_i, we have

$$\begin{aligned}&(Q-\epsilon)gain_{\mathcal{A}}(\rho) - adv(\rho)\\ &\quad=\; (Q-\epsilon)gain_i - adv_i + (Q-\epsilon)(x_i + b_i x_i) - (x_i + a_i x_i + b_i x_i)\\ &\quad\le\; [\Psi_i + Q - \epsilon - 1 + (Q-\epsilon-1)b_i - a_i]x_i \;=\; -\epsilon(1+b_i)x_i \;<\; 0.\end{aligned}$$

In the remaining case (C), $\mathcal{A}$ executes first x_i, then $a_i x_i$, and then $b_i x_i$. Using the formulas for a_i, b_i, Ψ_{i+1}, and the defining equation $Q^3 + Q^2 - 4Q + 1 = 0$, we have

$$
\begin{aligned}
&(Q-\epsilon)gain_{i+1} - adv_{i+1} \\
&\qquad \le (Q-\epsilon)gain_i - adv_i + (Q-\epsilon)(x_i + a_i x_i + b_i x_i) - (a_i x_i + 2b_i x_i) \\
&\qquad \le [\Psi_i + Q + (Q-1)a_i - (2-Q)b_i]x_i \;=\; b_i\Psi_{i+1}x_i \;=\; x_{i+1}\Psi_{i+1}.
\end{aligned}
$$

This completes the proof of Claim A.

Lemma 4.1 and Claim A imply that, for i large enough, we have $(Q-\epsilon)gain_i - adv_i \le \Psi_i x_i < (1 - Q + \epsilon)x_i$. For this i, if the game has not stopped earlier, the adversary ends it after stage i. Denoting by ϱ the sequence of all jobs (including the pending jobs x_i), we have

$$
\begin{aligned}
(Q-\epsilon)gain_{\mathcal{A}}(\varrho) - adv(\varrho) &= (Q-\epsilon)gain_i - adv_i + (Q-\epsilon)x_i - x_i \\
&< (1 - Q + \epsilon)x_i + (Q-\epsilon)x_i - x_i \;=\; 0.
\end{aligned}
$$

This completes the proof of the lemma, except that in the argument so far we assumed that the additive constant is 0. This assumption is easy to eliminate: For any given additive constant B, simply choose the initial job $x_1 \gg B$. The remainder of the proof is a simple modification of the presented argument. □

4.2 Upper Bound

We now present our Q-competitive algorithm for the 2-uniform case. Given that the 2-uniform case seems to be the most elementary case of unit job scheduling (without being trivial), our algorithm (and its analysis) is surprisingly difficult. Recall, however, that, as shown in [Bartal et al. 2004; Chin et al. 2006], any algorithm for this case with competitive ratio below $\sqrt{2}$ needs to use some information about the past. Further, when the adversary uses the strategy from Theorem 4.2, any Q-competitive algorithm needs to behave in an essentially unique way. Our algorithm was designed to match this optimal strategy, and then extended (by interpolation) to other adversarial strategies. Thus we suspect that the complexity of the algorithm is inherent in the problem and cannot be avoided.

We start with some intuitions. Let $\mathcal{A}$ be our online algorithm. Suppose that at time t we have one old pending job z, and two new pending jobs b, c with $b \ge c$. In some cases, the decision which job to execute is easy. If $c \ge z$, $\mathcal{A}$ can ignore z and execute b in the current step. If $z \ge b$, $\mathcal{A}$ can ignore c and execute z in the current step. If $c < z < b$, $\mathcal{A}$ faces a dilemma: it needs to decide whether to execute z or b. For $c = 0$, the choice is based on the ratio z/b. If z/b exceeds a certain threshold (possibly dependent on the past), we execute z, otherwise we execute b. Taking those constraints into account, and interpolating for arbitrary values of c, we can handle all cases by introducing a parameter η, $0 \le \eta \le 1$, and making the decision according to the following procedure:

Procedure $\textsc{Choose}_\eta$**:** If $z \ge \eta b + (1-\eta)c$ schedule z, otherwise schedule b.

To derive an online algorithm, say $\mathcal{A}$, we need to determine what values of η to use at each step. To this end, we examine the adversary strategy in the lower bound proof. Consider the limit case, when $i \to \infty$, and let $a_* = \lim_{i\to\infty} a_i = Q/(Q-1)$ and $b_* = \lim_{i\to\infty} b_i = Q/(Q-1)^2$.

Suppose that in the previous step two jobs z were issued. If the adversary now issues a single job a, then $\mathcal{A}$ needs to do the following: if $z \geq a/a_*$, execute z, and if $z \leq a/a_*$, then execute a. (The tie for $z = a/a_*$ can be broken either way.) Thus in this case we need to apply $\textsc{Choose}_\alpha$ with the threshold $\alpha = 1/a_* = (Q-1)/Q$.

Now, suppose that in the first step $\mathcal{A}$ executed z, so that in the next step a is pending. If the adversary now issues a single job b, then (assuming in the previous step the optimal value of $a \approx a_*$ was used) $\mathcal{A}$ must to do the following: if $a \geq b/b_*$, execute a, and if $a \leq b/b_*$, then execute b. Thus in this case we need to apply $\textsc{Choose}_\beta$ with the threshold $\beta = a_*/b_* = Q-1$.

Suppose that we execute a. In the lower-bound strategy, the adversary would now issue two jobs b in the next step, in which case we can use $\eta = \alpha$. But what happens if he issues a single job, say c? Calculations show that $\mathcal{A}$, in order to be Q-competitive, needs to use yet another parameter η in $\textsc{Choose}_\eta$. This parameter is not uniquely determined, but it must be at least $\gamma = (3-2Q)/(2-Q) > Q-1$. Further, it turns out that the same value γ can be used on subsequent single-job requests.

Our algorithm is derived from the above analysis: on a sequence of single-job requests in a row, use $\textsc{Choose}_\eta$ with parameter α in the first step, then β in the second step, and γ in all subsequent steps. In general, of course, two jobs can be issued at each step (or more, but only the two heaviest jobs need to be considered). We think of an algorithm as a function of several arguments. The values of this function on the boundary are determined from the optimal adversary strategy, as explained above. The remaining values are obtained through interpolation.

We now give a formal description of our algorithm. Let

$$\alpha = \frac{Q-1}{Q} \approx 0.27, \qquad \beta = Q-1 \approx 0.38, \qquad \gamma = \frac{3-2Q}{2-Q} \approx 0.39,$$

$$\lambda(\xi) = \min\left\{1, \frac{\xi-\alpha}{\beta-\alpha}\right\}, \qquad \delta(\mu,\xi) = \mu\alpha + (1-\mu)[\beta + (\gamma-\beta)\lambda(\xi)],$$

where $0 \leq \mu \leq 1$ and $\alpha \leq \xi \leq \gamma$. Note that the function $\lambda(\xi)$ increases from 0 to 1 as ξ increases from α to β, and is equal to 1 for $\beta \leq \xi \leq \gamma$. For parameters μ and ξ withing their ranges, the function $\delta(\mu,\xi)$ satisfies $\alpha \leq \delta(\mu,\xi) \leq \gamma$. Further, we have $\delta(1,\xi) = \alpha$, $\delta(0,\xi) \geq \beta$ for any ξ, and $\delta(0,\alpha) = \beta$, $\delta(0,\beta) = \delta(0,\gamma) = \gamma$.

Algorithm $\textsc{Switch}$. Without loss of generality, we assume that at each step exactly two jobs are released. If more jobs are released, consider only the two heaviest jobs. If fewer jobs are released, create dummy jobs with weight 0.

Fix a time step t. Let b, c (where $b \geq c$) be the two jobs released at time t, and u, v (where $u \geq v$) be the two jobs released at time $t-1$. (Initially, at $t = 0$, let $u = v = 0$.)

We distinguish two cases. If $u = v$, or if u was scheduled at time $t-1$, then run the job selected by $\textsc{Choose}_\alpha$. (Note that this includes the case $t = 0$.) Otherwise, denoting by ξ the parameter of $\textsc{Choose}_\xi$ executed at time $t-1$, run the job selected by $\textsc{Choose}_\eta$, for $\eta = \delta(v/u, \xi)$.

THEOREM 4.3. *Algorithm* SWITCH *is* Q*-competitive for the* 2*-uniform case, where* $Q \approx 1.377$ *is the largest root of* $Q^3 + Q^2 - 4Q + 1 = 0$.

The proof of the theorem is included in the appendix.

APPENDIX

A. THE PROOF OF THEOREM 4.3

In this section we prove Theorem 4.3. The analysis of the 2-uniform case is based on the potential function argument. We define below a potential function Φ that maps all possible configurations into real numbers, and we prove a bound on the amortized cost of the algorithm in each step.

We fix an adversary schedule to be a canonical optimal schedule. Thus if the adversary schedules both jobs released at the same time, then the heavier one is scheduled first.

The configuration at time t is specified by the parameter ξ of CHOOSE$_\xi$ used at time $t-1$, the jobs $u \geq v$ released at time $t-1$, and the pending jobs $x, y \in \{u, v\}$ of the algorithm and the adversary, respectively, at time t. (For $t = 0$ we set $u = v = x = y = 0$ and $\xi = \alpha$.) We denote the potential at time t by $\Phi_{xy}(u, v, \xi)$, and define it as follows:

$$
\begin{aligned}
E &= 2 - Q^2 \;=\; \alpha(Q-1) \;\approx\; 0.104,\\
G &= 2(Q-1) - 1/Q \;\approx\; 0.028,\\
\Phi_{vy}(u, v, \xi) &= y - Qv,\\
\Phi_{uu}(u, v, \xi) &= E \cdot \lambda(\xi)(u + v),\\
\Phi_{uv}(u, v, \xi) &= (Q-1)v - (G \cdot \lambda(\xi) + 1/Q)u.
\end{aligned}
$$

We now consider a single step at time t. Let b and c be the jobs released at time t, where $b \geq c$. Let η denote the parameter of CHOOSE used at time t, and let $x', y' \in \{b, c\}$ denote the pending jobs of the algorithm and the adversary at $t+1$, respectively. The potential at time $t+1$ is $\Phi_{x'y'}(b, c, \eta)$; we refer to it as the 'new potential' as opposed to the 'old potential' $\Phi_{xy}(u, v, \xi)$ at time t. The rest of this section is devoted to the proof of the following inequality:

$$
\Gamma \;=\; \Phi_{xy}(u, v, \xi) + Q \cdot \Delta gain_{\text{SWITCH}} - \Delta adv - \Phi_{x'y'}(b, c, \eta) \;\geq\; 0, \tag{2}
$$

where $\Delta gain_{\text{SWITCH}}$ and Δadv are the gains of the algorithm and the adversary at time t. It is sufficient to prove inequality (2), because the Q-competitiveness of the algorithm follows immediately from (2) by summation over all times t and observing that $\Phi_{xy}(0, 0, \xi) = 0$, i.e., the potential is zero on configurations with no old pending jobs, which includes the initial and final configurations.

We are now ready to prove (2) by a case analysis. During the proof, we need to verify a number of relations between the constants we have defined so far. In most cases a rough calculation based on the numerical values given above is sufficient due to some slack; we explicitly mention the cases when the relations are tight.

<u>Case 1</u>: Suppose that v is pending for SWITCH at time t. Thus the old potential is $\Phi_{vy}(u, v, \xi) = y - Qv$, and SWITCH applies CHOOSE$_\alpha$ at time t. This also means that $\lambda(\eta) = \lambda(\alpha) = 0$.

Case 1.1: If $v < \alpha b + (1-\alpha)c$ then SWITCH schedules b and has $x' = c$ pending at time $t+1$.

If the adversary schedules y, it has $y' = b$ pending at time $t+1$, the new potential is $\Phi_{cb}(b,c,\alpha) = b - Qc$, and we get

$$\Gamma \;=\; (y - Qv) + Qb - y - (b - Qc) \;=\; (Q-1)b + Qc - Qv \;\geq\; 0,$$

where the last inequality follows from the case condition, after substituting $\alpha = (Q-1)/Q$.

Otherwise, the adversary schedules b, it has $y' = c$ pending at time $t+1$, the new potential is $\Phi_{cc}(b,c,\alpha) = (1-Q)c$, and using $y \geq v$ we get

$$\Gamma \;\geq\; (v - Qv) + Qb - b - (1-Q)c \;=\; (Q-1)(b+c) - (Q-1)v \;\geq\; 0,$$

where the last inequality follows from the case condition, after substituting $\alpha = (Q-1)/Q$.

Case 1.2: If $v \geq \alpha b + (1-\alpha)c$ then CHOOSE$_\alpha$ schedules v and has $x' = b$ pending.

If the adversary schedules y, it has $y' = b$ pending at time $t+1$, the new potential is $\Phi_{bb}(b,c,\alpha) = 0$, and we get

$$\Gamma \;=\; (y - Qv) + Qv - y \;=\; 0.$$

Otherwise, the adversary schedules b, it has $y' = c$ pending at time $t+1$, the new potential is $\Phi_{bc}(b,c,\alpha) = (Q-1)c - b/Q$, and using $y \geq v$ we get

$$\Gamma \;\geq\; (v - Qv) + Qv - b - ((Q-1)c - b/Q) \;=\; v - \alpha b - (Q-1)c \;\geq\; 0,$$

where the last inequality follows from $Q - 1 < 1 - \alpha$ and the case condition.

Case 2: In this case $u > 0$ is the pending job for SWITCH at time t, and at time t SWITCH applies CHOOSE$_\eta$, where $\eta = \delta(v/u, \xi)$. The old potential is $\Phi_{uy}(u,v,\xi)$. To reduce the number of subcases, we first note that

$$\Phi_{uu}(u,v,\xi) \;\geq\; \Phi_{uv}(u,v,\xi). \tag{3}$$

Indeed, after substituting, this is equivalent to $((E+G)\lambda(\xi) + 1/Q)u \geq (Q - 1 - E\cdot\lambda(\xi))v$. Since $\lambda(\xi) \geq 0$ and $u \geq v$, it is sufficient to verify that $1/Q > Q - 1$, which is true.

Since Γ is a linear function of job weights and, when the job weights are rescaled, η as well as other coefficients do not change, we may assume that $u = 1$. It is also convenient to define $l = \lambda(\xi)$. Recapitulating the definitions, we have

$$\begin{aligned} l \;&=\; \lambda(\xi) \;=\; \min\{1, (\xi - \alpha)/(\beta - \alpha)\}, \\ \eta \;&=\; \eta(v,l) \;=\; v\alpha + (1-v)(\beta + (\gamma - \beta)l) \;=\; \delta(v,\xi). \end{aligned}$$

By the properties of the function $\lambda(\xi)$, we have $0 \leq l \leq 1$. The function $\eta(v,l)$ is non-increasing in v and non-decreasing in l in the whole domain of v and l. Furthermore, $\alpha \leq \eta(v,l) \leq \gamma$, $\eta(1,l) = \alpha$, and $\eta(0,l) \geq \beta$.

In each case of the analysis we need to minimize a linear function of b and c subject to $0 \leq c \leq b$ and $\eta b + (1-\eta)c = 1$. Since the feasible domain is a line segment (see Figure 3), the minimum must be attained at one of the endpoints

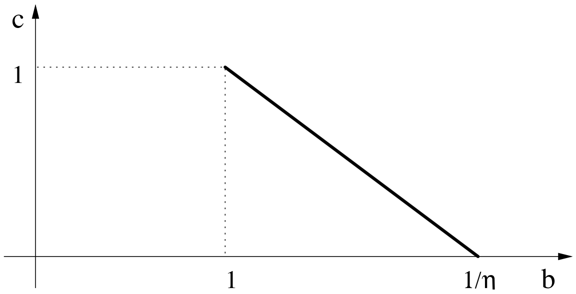

Fig. 3. The domain of b, c in Case 2 (thick line).

which are $(b, c) = (1, 1)$ and $(b, c) = (1/\eta, 0)$. Thus the minimum can be found by comparing these two values of the investigated linear function.

Case 2.1: $1 < \eta b + (1-\eta)c$ and thus SWITCH schedules b.

Case 2.1.1: The adversary schedules $y = u = 1$. Using $v \geq 0$, we bound the old potential as $\Phi_{uy}(u, v, \xi) = \Phi_{uu}(u, v, \xi) \geq E \cdot l$. The new potential is $\Phi_{cb}(b, c, \eta) = b - Qc$ and we get

$$\begin{aligned}
\Gamma &\geq E \cdot l + Qb - 1 - (b - Qc) \\
&= E \cdot l - 1 + (Q-1)b + Qc \qquad (4) \\
&\geq E \cdot l - 1 + (Q-1)/\eta \qquad (5)
\end{aligned}$$

The last inequality is justified as follows: expression (4) is a linear function of b, c, increasing in both b and c. From the case condition and $b \geq c \geq 0$ it follows that (4) decreases when b and/or c are decreased until $\eta b + (1-\eta)c = 1$, and under this constraint, $(Q-1)b + Qc$ is minimized for $b = 1/\eta$, $c = 0$. At the other endpoint of the feasible region, i.e., at $b = c = 1$, the value is larger, since $2Q - 1 > (Q-1)/\alpha \geq (Q-1)/\eta$.

Now (5) is minimized for $v = 0$, because $\eta = \delta(v, \xi)$ is decreasing in v. We substitute $v = 0$ in (5), multiply by $\eta(0, l)$, and substitute the definition of $\eta(0, l)$ to obtain

$$\begin{aligned}
\eta(0, l) \cdot \Gamma &\geq E \cdot l(\beta + (\gamma - \beta)l) - \beta - (\gamma - \beta)l + Q - 1 \\
&\geq E \cdot l\beta - \beta - (\gamma - \beta)l + Q - 1 \\
&\geq 0.
\end{aligned}$$

The final inequality holds since $\beta = Q - 1$ and $E \cdot \beta > \gamma - \beta$. This holds, since $E \cdot \beta > 0.037$ and $\gamma - \beta < 0.02$, including the rounding errors. Alternatively, substituting the definitions of E, β, and γ, the inequality reduces to a degree-4 polynomial inequality in Q which, using the definition of Q, can be reduced to degree-2 inequality $7Q^2 - 6Q - 5 > 0$, that can be verified using again the definition of Q.

Case 2.1.2: The adversary schedules $y = v$. Using $l \leq 1$ and the definition of G, the old potential is bounded by $\Phi_{uy}(u, v, \xi) = \Phi_{uv}(u, v, \xi) = (Q-1)v - G \cdot l - 1/Q \geq$

$(Q-1)v - 2(Q-1)$. The new potential is $\Phi_{cb}(b,c,\eta) = b - Qc$. We bound the linear function of b and c exactly as in the previous case to obtain

$$\begin{aligned}\Gamma &= (Q-1)v - 2(Q-1) + Qb - v - (b - Qc) \\ &= -(2-Q)v - 2(Q-1) + (Q-1)b + Qc \\ &\geq -(2-Q)v - 2(Q-1) + (Q-1)/\eta.\end{aligned}$$

We now notice that $\eta = \eta(v,l)$ is increasing with l, thus it is sufficient to substitute the value of η for $l = 1$. In this case we get after multiplying by $\eta(v,1)$ and substituting its value

$$\eta(v,1)\cdot\Gamma \geq -(\alpha v + (1-v)\gamma)((2-Q)v + 2(Q-1)) + (Q-1).$$

For $v = 1$, the right-hand side is equal to 0. To conclude that $\Gamma \geq 0$, it is sufficient to show that the right-hand side decreases for $v \in [0,1]$. It is a convex quadratic function (as $\gamma > \alpha$), thus it is sufficient to verify that its derivative at $v = 1$ is at most 0. The derivative is $(\gamma-\alpha)((2-Q)v + 2(Q-1)) - (\alpha v + (1-v)\gamma)(2-Q)$, which at $v = 1$ equals $\gamma Q - 2\alpha < 0$.

Case 2.1.3: The adversary schedules b. Using (3), $v \geq 0$, $l \leq 1$, and the definition of G the old potential is bounded by $\Phi_{uy}(u,v,\xi) \geq \Phi_{uv}(u,v,\xi) \geq -G - 1/Q = -2(Q-1)$. The new potential is $\Phi_{cc}(b,c,\eta) = (1-Q)c$. We have

$$\begin{aligned}\Gamma &\geq -2(Q-1) + Qb - b - (1-Q)c \\ &= (Q-1)(b + c - 2) \\ &\geq 0.\end{aligned}$$

To justify the last inequality, note that, given the case constraint, $b+c$ is minimized at $b = c = 1$, as the value at $b = 1/\eta$, $c = 0$ is larger, since $2 < 1/\gamma \leq 1/\eta$.

Case 2.2: Suppose that $1 \geq \eta b + (1-\eta)c$. Then SWITCH executes $u = 1$.

Case 2.2.1: The adversary schedules $y = u = 1$. The old potential is bounded by $\Phi_{uy}(u,v,\xi) = \Phi_{uu}(u,v,\xi) \geq 0$ and the new potential is $\Phi_{bb}(b,c,\eta) = E\cdot\lambda(\eta)(b+c)$. We get

$$\begin{aligned}\Gamma &\geq Q - 1 - E\cdot\lambda(\eta)(b+c) \\ &\geq Q - 1 - E\cdot\lambda(\eta)/\eta.\end{aligned}$$

The last inequality follows since $b+c$ is maximized when $\eta b + (1-\eta)c = 1$, using the case condition. Under this restriction, it is maximized when $b = 1/\eta$, $c = 0$, as the value for $b = c = 1$ is smaller because $1/\eta \geq 1/\gamma > 2$.

Using the definition of λ, the value of $\lambda(\eta)/\eta$ is maximized for $\eta = \beta$, where $\lambda(\eta)/\eta = 1/\beta$. Thus $\Gamma \geq Q - 1 - E/\beta > 0$.

Case 2.2.2: The adversary schedules $y = v$. Using $l \leq 1$ and the definition of G, the old potential is bounded by $\Phi_{uy}(u,v,\xi) = \Phi_{uv}(u,v,\xi) = (Q-1)v - G\cdot l - 1/Q \geq (Q-1)v - 2(Q-1)$. The new potential is $\Phi_{bb}(b,c,\eta) = E\cdot\lambda(\eta)(b+c)$. We bound the linear function of b and c exactly as in the previous case to obtain

$$\Gamma \geq (Q-1)v - 2(Q-1) + Q - v - E\lambda(\eta)(b+c)$$

$$\geq\ (2-Q)(1-v) - \frac{E \cdot \lambda(\eta)}{\eta}$$
$$\geq\ (2-Q)(1-v) - \frac{E(\eta-\alpha)}{\alpha(\beta-\alpha)}, \tag{6}$$

where the last inequality follows from $\eta \geq \alpha$ and the definition of λ. The right-hand side of (6) is linear in v, since $\eta = \eta(v,l)$ is a linear function of v. Thus it is sufficient to verify that (6) is non-negative for $v \in \{0,1\}$. For $v = 1$, it is equal to 0. For $v = 0$, we use $\eta \leq \gamma$ and the whole expression is at least

$$2 - Q - \frac{E(\gamma-\alpha)}{\alpha(\beta-\alpha)} > 0.$$

To verify the last inequality numerically, note that $E/\alpha = Q - 1 < 0.4$, so it is sufficient to check that $(\gamma-\alpha)/(\beta-\alpha) < 1.5$. Alternatively, the inequality can be verified by substituting the definitions of the parameters in terms of Q and reducing it to $5Q^2 - 12Q + 7 < 0$, which holds by the definition of Q.

<u>Case 2.2.3</u>: The adversary schedules b. Using inequality (3), the old potential is bounded by $\Phi_{uy}(u,v,\xi) \geq \Phi_{uv}(u,v,\xi) = (Q-1)v - G\cdot l - 1/Q$. The new potential is $\Phi_{bc}(b,c,\eta) = (Q-1)c - (G\cdot\lambda(\eta) + 1/Q)b$. We have

$$\Gamma\ \geq\ (Q-1)v - G\cdot l - 1/Q + Q - (1 - G\cdot\lambda(\eta) - 1/Q)b - (Q-1)c.$$

The expression on the right-hand side is a linear function of b and c. We claim that it is minimized at $b = 1/\eta$, $c = 0$. Indeed, subtracting its value at $b = 1/\eta$, $c = 0$, from the value at $b = c = 1$, we get

$$(1 - G\cdot\lambda(\eta) - 1/Q)(1/\eta - 1) - (Q-1)\ \geq\ (1 - G - 1/Q)(1/\gamma - 1) - (Q-1)\ =\ 0,$$

where we use $\lambda(\eta) \leq 1$ and $\eta \leq \gamma$ in the inequality, and the last equality follows from the definitions of G and γ. Thus, after substituting $b = 1/\eta$ and $c = 0$, and multiplying by η,

$$\eta\cdot\Gamma\ \geq\ \eta\cdot((Q-1)v - G\cdot l - 1/Q + Q) - 1 + G\cdot\lambda(\eta) + 1/Q. \tag{7}$$

We want to show that the right-hand side of (7) is non-negative.

If $\eta \geq \beta$, then $\lambda(\eta) = 1$, and by the definitions of G and η, the right-hand side of (7) is equal to

$$[v\alpha + (1-v)(\beta + (\gamma-\beta)l)]((Q-1)v - G\cdot l - 1/Q + Q) - (3 - 2Q).$$

This is a concave quadratic function in v, so it is sufficient to verify that it is non-negative for $v \in \{0,1\}$. For $v = 1$, the function is decreasing with l, so it is minimized at $l = 1$ and the value is $\alpha - (3 - 2Q) > 0$. For $v = 0$, the function is a concave quadratic function in l, so it is sufficient to verify that it is non-negative for $l \in \{0,1\}$. For $l = 0$ the value is $\beta(-1/Q + Q) - (3 - 2Q)$, and for $l = 1$ it is $\gamma(2 - Q) - (3 - 2Q)$. Both values are equal to 0, by the definitions of β, γ, and Q.

It remains to verify that (7) is non-negative when $\eta \leq \beta$. In this case, the right-hand side of (7) is equal to

$$\eta\cdot((Q-1)v - G\cdot l - 1/Q + Q) - 1 + 1/Q + G\cdot(\eta-\alpha)/(\beta-\alpha). \tag{8}$$

For each l, $\eta(v, l)$ is continuous and decreasing in v from $\eta(0, l) \geq \beta$ to $\eta(1, l) = \alpha < \beta$, so there is unique $v = v_l$ for which $\eta(v, l) = \beta$. The expression (8) is again a concave quadratic function in v, and we know that it is non-negative at $v = v_l$ from the analysis of the previous case. As in this sub-case we have $v_l \leq v \leq 1$, it remains to verify that (8) is non-negative for $v = 1$. In this case its value is $\alpha(2Q - 1 - G \cdot l - 1/Q) - 1 + 1/Q \geq 0$, using $l \leq 1$ and the definition of G.

We have now examined all cases, completing the proof of inequality (2), and thus also the proof of Theorem 4.3.

REFERENCES

Andelman, N., Mansour, Y., and Zhu, A. 2003. Competitive queueing policies in QoS switches. In *Proc. 14th Symp. on Discrete Algorithms (SODA)*. ACM/SIAM, 761–770.

Bansal, N., Fleischer, L., Kimbrel, T., Mahdian, M., Schieber, B., and Sviridenko, M. 2004. Further improvements in competitive guarantees for QoS buffering. In *Proc. 31st International Colloquium on Automata, Languages, and Programming (ICALP)*. Lecture Notes in Comput. Sci., vol. 3142. Springer, 196–207.

Bartal, Y., Chin, F. Y. L., Chrobak, M., Fung, S. P. Y., Jawor, W., Lavi, R., Sgall, J., and Tichý, T. 2004. Online competitive algorithms for maximizing weighted throughput of unit jobs. In *Proc. 21st Symp. on Theoretical Aspects of Computer Science (STACS)*. Lecture Notes in Comput. Sci., vol. 2996. Springer, 187–198.

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

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

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

Chrobak, M., Jawor, W., Sgall, J., and Tichý, T. 2004. Improved online algorithms for buffer management in QoS switches. In *Proc. 12th European Symp. on Algorithms (ESA)*. Lecture Notes in Comput. Sci., vol. 3221. Springer, 204–215.

Englert, M. and Westerman, M. 2007. Considering suppressed packets improves buffer management in QoS switches. In *Proc. 18th Symp. on Discrete Algorithms (SODA)*. ACM/SIAM, 209–218.

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

Kesselman, A., Lotker, Z., Mansour, Y., Patt-Shamir, B., Schieber, B., and Sviridenko, M. 2001. Buffer overflow management in QoS switches. In *Proc. 33rd Symp. Theory of Computing (STOC)*. ACM, 520–529.

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

Kesselman, A., Mansour, Y., and van Stee, R. 2003. Improved competitive guarantees for QoS buffering. In *Proc. 11th European Symp. on Algorithms (ESA)*. Lecture Notes in Comput. Sci., vol. 2832. Springer, 361–372.

Kesselman, A., Mansour, Y., and van Stee, R. 2005. Improved competitive guarantees for QoS buffering. *Algorithmica 43*, 63–80.

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

Li, F., Sethuraman, J., and Stein, C. 2007. Better online buffer management. In *Proc. 18th Symp. on Discrete Algorithms (SODA)*. ACM/SIAM, 199–208.

Received January 2006; revised October 2006; accepted October 2006