A ϕ-Competitive Algorithm for Collecting Items with Increasing Weights from a Dynamic Queue

Marcin Bienkowski[a,2], Marek Chrobak[c,1], Christoph Dürr[d,3], Mathilde Hurand[e,3], Artur Jeż[a,2], Łukasz Jeż[a,b,2], Grzegorz Stachowiak[a,2]

[a] *Institute of Computer Science, University of Wrocław, 50-383 Wrocław, Poland.*
[b] *Institute of Mathematics, Academy of Sciences of the Czech Republic, Žitná 25, 115 67 Praha 1, Czech Republic*
[c] *Department of Computer Science, University of California, Riverside, CA 92521, US.*
[d] *CNRS, LIP6, Université Pierre et Marie Curie, 75252 Paris Cedex 05, France.*
[e] *Google, 8002 Zürich, Switzerland.*

Abstract

The *bounded-delay packet scheduling (or buffer management) problem* is to schedule transmissions of packets arriving in a buffer of a network link. Each packet has a deadline and a weight associated with it. The objective is to maximize the weight of packets that are transmitted before their deadlines, assuming that only one packet can be transmitted in one time step. Online packet scheduling algorithms have been extensively studied. It is known that no online algorithm can achieve competitive ratio better than $\phi \approx 1.618$ (the golden ratio), while the currently best upper bound on the competitive ratio is $2\sqrt{2}-1 \approx 1.824$. Closing the gap between these bounds remains a major open problem.

The above mentioned lower bound of ϕ uses instances where item weights increase exponentially over time. In fact, all lower bounds for various versions of buffer management problems involve instances of this type. In this paper, we design an online algorithm for packet scheduling with competitive ratio ϕ when packet weights are increasing, thus matching this lower bound. Our algorithm applies, in fact, to a much more general version of packet scheduling, where only relative order of the deadlines is known, not their exact values.

Keywords: online algorithms, competitive analysis, buffer management, packet scheduling
2000 MSC: 68W01, 68W05, 68W20, 68W40, 68Q25

1. Introduction

The *bounded-delay packet scheduling (or buffer management) problem* is to schedule transmissions of packets arriving at a network link. Arriving packets are stored in a buffer whose size is unbounded. Each packet has a deadline and a weight associated with it. The time is discrete and only one packet can be transmitted in one time step. The objective is to maximize the weight of packets that are transmitted before their deadlines expire. Online packet scheduling algorithms have been extensively studied in the literature [1, 2, 3, 4, 5, 6, 7, 8, 9]. It has been known for some time that no online algorithm can achieve competitive ratio better than $\phi \approx 1.618$, the golden ratio [1, 3]. The greedy algorithm, which always transmits the maximum weight pending packet, is 2-competitive [5, 6] and, after a sequence of improvements, the best upper bound on the competitive ratio has been now reduced to $2\sqrt{2}-1 \approx 1.828$ [4]. Online algorithms with ratio ϕ have been developed for some special cases, including the 2-bounded case [6], where each packet must be transmitted within at most two steps, or the more general case of agreeable deadlines [8], where the items are released in order of non-decreasing deadlines. For the general case, closing the gap between these bounds remains a major open problem in the area of online buffer management algorithms.

In the companion paper [10], we introduced a more general version of packet scheduling, where only relative order of the deadlines is known, not their exact values. This model is motivated by packet scheduling problems in networks with tiered traffic, competition with other traffic streams, interference on wireless links, or other situations where access to the communication channel is intermittent and unpredictable. In [10], the model is described as follows: we have a dynamic queue S that stores items with weights. At each time step, some items can be inserted into S at arbitrary locations, and a prefix of S can be deleted. We can collect one item from S at each step, with the restriction that each item can be collected at most once. (The collected item stays in the queue. Put another way, we actually only collect the value of the item.) The objective is to maximize the total weight of collected items. For brevity, we will refer to this problem as the *weighted item collection* problem.

In the case of packet scheduling, items represent packets, with the queue S containing all packets that have been released but have not yet expired, ordered by their deadlines. As new packets are released they are inserted into S in appropri-

[1]Research supported by NSF grants OISE-0340752, CCF-0729071 and CCF-1217314.

[2]Supported by MNiSW grants N N206 368839, 2010–2013.

[3]Research supported by ANR Alpage and ANR Netoc. Most of the work was carried out while the author was at LIX Ecole Polytechnique.

ate locations, consistent with the ordering. At each step the expired packets are removed from S. Due to their ordering, this will indeed result in removing a prefix of S. Collecting an item represents transmitting the corresponding packet. (Note that the buffer will then contain the items in S that have not been transmitted by the algorithm.) The difference between the two problems can be delineated as follows: while in packet scheduling the arrivals of packets are unpredictable, once a packet arrives its exact deadline is revealed. In the item collection problem, both arrival and expiration times are not known; we only know that the items will expire in the order determined by the queue.

Since packet scheduling is a special case, a lower bound of ϕ applies to the weighted item collection problem as well. In fact, in our setting the proof can be considerably simplified, as we will demonstrate shortly. A stronger lower bound of ≈ 1.633 was shown in [10]. As for upper bounds, the greedy algorithm generalizes naturally to the weighted item collection problem and it remains 2-competitive. The main result of [10] is a better upper bound of ≈ 1.89.

In weighted item collection, the online algorithm faces the trade-off between the number of items it can collect and their total weight. To collect more items, it is better to collect those early in the queue, since they will be deleted first. On the other hand, items later in the queue may have higher weights, but if we collect those items we may end up with fewer items overall, because the earlier items can get deleted. This trade-off is illustrated by the example in Figure 1, where an online algorithm A collects items 25 and 35. In hindsight, in the first step it would have been better to collect item 40, since it was deleted at the same time as 25, and in the second step it would have been better to collect 20, since then A could still collect 35 in the next step. In this example, A's gain is $25+35=60$, while the optimum gain is $40+20+35=95$.

Let us consider another example, which is actually an adversary strategy that proves the lower bound of ϕ on the competitiveness of any online algorithm. Suppose that the queue initially contains two items, of weights 1 and ϕ, in this order. If the algorithm collects the item of weight 1, then let both items be deleted after the first step, in which case the optimum gain is ϕ. If the algorithm collects the item of weight ϕ, then let only the first item be deleted after the first step. As a result, the algorithm's gain is ϕ, while the optimum gain is $1+\phi=\phi^2$. In both cases, the optimum gain is ϕ times the algorithm's gain, proving a lower bound of ϕ on the competitive ratio. Note that this lower bound strategy cannot be emulated in the packet scheduling model, since the second item is released at the beginning but its expiration time depends on the algorithm's choice in the first step. (The lower bound of ϕ for packet scheduling in [1, 3] is much more involved.)

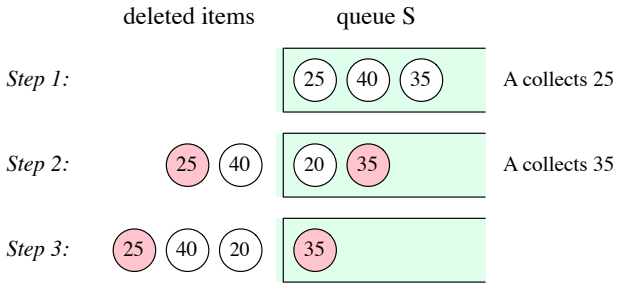

Figure 1: An example of an online algorithm A collecting items from a dynamic queue. Items are identified by their weights. Collected items are shaded. Initially the queue contains items 25, 40 and 35. In the first step, A collects item 25. After the first step items 25, 40 are deleted and item 20 is inserted right before item 35. In the second step, A collects item 35. After the second step item 20 is deleted. In the last step (not shown) item 35 will be deleted, completing the computation.

Our results. We design an online algorithm for weighted item collection with competitive ratio ϕ for instances where item weights are increasing with respect to their ordering in the queue. (See the formal definition in the next section.) Since the lower bound presented above uses an instance with increasing weights, this establishes a tight bound of ϕ for such instances.

The assumption about increasing weights is motivated by the lower bound techniques for bounded-delay packet scheduling [1, 3, 5], which use instances where the weights of packets increase with respect to their deadlines. (In fact, these weights increase exponentially.) Our result implies that for such instances the ratio of ϕ is tight. Lower bound strategies that use increasing weights are commonly used in various packet buffering models; other examples include lower bounds for packet scheduling for 2-bounded instances and 2-uniform instances [11], or lower bounds for buffer management with bounded-size buffers [1, 6, 12].

The significance of our contribution can be summarized as follows. If there exists a ϕ-competitive algorithm for packet scheduling, the ideas from our work should be useful in designing such an algorithm. On the other hand, if the optimal ratio for packet scheduling is larger than ϕ, then our proof shows that the lower bound construction for a better bound will require new ideas, based on a non-increasing sequence. It could be possible, for example, to adapt the approach from [10] that uses a bitonic (U-shaped) sequence. We also point out that for decreasing sequences the obvious greedy algorithm computes an optimum solution.

Note. A preliminary version of this work appeared in the Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms (SODA'09) [13]. Other results from [13] have been published separately in the companion paper [10].

2. Preliminaries

We adapt the notation and terminology from [10]. At any time step, the items currently in the queue S (that is, the items already inserted but not yet deleted) are referred to as *active*. The queue ordering relation is denoted "$\lhd$", that is $a \lhd b$ means that a precedes b in the queue. Deleted items cannot be reinserted into the queue, so this relation is well defined. We can extend "$\lhd$" to a linear order on the whole instance by letting $a \lhd b$ if a was deleted from the queue before b was inserted. This is summarized in the following observation.

Fact 1. At any time, relation "$\lhd$" is a total order on all already released items, with the currently active items forming a suffix of this ordering.

The weight of an item x is denoted by w_x or $w(x)$. We extend this notation to sets: the total weight of a set X of items is denoted by $w(X)$. To simplify the arguments we assume that all weights are different; otherwise we can modify the weight comparison relation so that for any two items of equal weight the one later in the queue is considered heavier. We will say that an instance of the item collection problem is *monotone* if any two items a, b with $a \lhd b$ satisfy $w_a < w_b$. (By our assumption that there are no two items of the same weight the latter is equivalent to $w_a \leq w_b$.) In particular, at any time step the weight of the active items increases with respect to their position in the queue. Further, all already deleted items have smaller weights than all active items.

We use standard terminology from competitive analysis. An algorithm A is called *online* if at each step its decision as to what item from S to collect is independent of future queue updates. A is called *R-competitive* if its gain on any instance I is at least the optimum gain on I divided by R. The *competitive ratio* of A is the smallest R for which A is R-competitive.

An item is called *pending* for an online algorithm A at a given step if it is active but not yet collected by A.

It is easy to show that, without loss of generality, optimal (the adversary's) solutions satisfy the following property:

Earliest-Expiration-First (EEF) Property: For any two active items $a \lhd b$, if the adversary collects b at the current time step, then the adversary will not collect a in the future.

If such b is collected, we will say that the adversary *forfeits* a. We say that an item a is *pending for the adversary* at a certain time step if it can be collected later by the adversary satisfying the EEF property. In other words, this a is active, and neither collected nor forfeited by the adversary. Note that the concept of pending items is different for online algorithms and the adversary.

3. Algorithm MARK&PICK

In this section, we present an online algorithm MARK&PICK that is ϕ-competitive for monotone instances, namely when item weights are increasing with respect to the queue order, as defined in the previous section. Recall that all weights are assumed to be different.

We think of each time step as consisting of two parts. In the first part a prefix of the queue is deleted and some items are inserted into the queue. In the second part, the algorithm can collect an item.

Algorithm 1 MARK&PICK (single step)

```
/* update queue */
if there is no pending item then skip this step
m ← the heaviest unmarked item (not necessarily active)
mark m
collect the earliest pending item ℓ with w_ℓ ≥ w_m/φ
```

It is not obvious that this algorithm is well defined. A formal proof, given in the next section, involves defining a number of invariants and showing that they are preserved throughout the computation. To gain some intuition, assume that in all previous steps the items m and ℓ were well-defined. Thus, at each previous step, MARK&PICK marked a single item and collected a single item. The consequence of this is that at the beginning of the current step the number of marked items is equal to the number of collected items. Thus, if there is a pending item then there must also be an unmarked item in the instance (possibly already deleted from the queue). This implies that m is well-defined. Furthermore, if m is pending, then m itself is a candidate for ℓ, so ℓ is also well-defined in this case. However, the proof of the existence of ℓ when m is not pending is more involved.

3.1. Dominance Relation

Let U, V be two sets of real numbers. We say that V *dominates* U, denoted $V \succeq U$, if and only if there is an injection $f : U \mapsto V$ such that $f(u) \geq u$ for all $u \in U$. From the definition, $V \succeq U$ implies that $|V| \geq |U|$. (To emphasize, we do not require that $|U| = |V|$.)

For a set X and an element x, we write $X \cup x = X \cup \{x\}$, for simplicity. We now show that, under some conditions, adding u to U and v to V may preserve the dominance relation $V \succeq U$. In the lemma below, we assume that $\max(\emptyset) = -\infty$.

Lemma 1. Let Π be a finite set of real numbers and $U, V \subseteq \Pi$, where $V \succeq U$. Consider two numbers $u \in \Pi \setminus U$ and $v \in \Pi \setminus V$. If $v \geq u$ or $v = \max(\Pi \setminus V)$, then $V \cup v \succeq U \cup u$.

Proof. If $v \geq u$ then the claim clearly holds: it is enough to take the injection $f : U \mapsto V$ showing that $V \succeq U$ and extend it to u, by putting $f(u) = v$. So, in the following, we assume that $v = \max(\Pi \setminus V) < u$.

For any $X \subseteq \Pi$ and $x \in \Pi$, let $|X|_{\geq x} = |\{z : z \in X, z \geq x\}|$ denote the number of elements in X that are at least x. It is easy to see that $V \succeq U$ if and only if $|V|_{\geq x} \geq |U|_{\geq x}$ for all $x \in \Pi$. We use this characterization in the proof.

For $x \leq v$, it holds that

$$|V \cup v|_{\geq x} = |V|_{\geq x} + 1 \geq |U|_{\geq x} + 1 = |U \cup u|_{\geq x} \ .$$

For $v < x$ note that, as $v = \max(\Pi \setminus V)$, it holds that $|V|_{\geq x} = |\Pi|_{\geq x}$ and so

$$|V \cup v|_{\geq x} = |V|_{\geq x} = |\Pi|_{\geq x} \geq |U \cup u|_{\geq x} \ .$$

This concludes the proof. □

It will be convenient to extend the dominance relation to sets of items A, B: We will write $A \succeq B$ if the same relation holds for the weights in A and B, that is, $\{w_a : a \in A\} \succeq \{w_a : a \in B\}$. Generalizing it further, for any real numbers α and β, we write $\alpha A \succeq \beta B$ if $\{\alpha w_a : a \in A\} \succeq \{\beta w_a : a \in B\}$.

3.2. Correctness

We introduce several dynamic sets of items that will be used in the analysis. Let M_t, L_t and Z_t denote, respectively, the sets of items marked by MARK&PICK, collected by MARK&PICK, and collected by the adversary up to and including step t. Whenever we need to refer to specific items from these sets, m_t, ℓ_t and z_t denote, respectively, the item m marked by MARK&PICK, the item ℓ collected by MARK&PICK, and the item collected by the adversary in step t. Note that for some steps t each of items ℓ_t, m_t, or z_t may be undefined.

Lemma 2. At each time t:

(i) if there is an item pending for MARK&PICK, then there is an unmarked item; further, if m_t is the heaviest unmarked item then there is a pending item ℓ_t such that $\phi w(\ell_t) \geq w(m_t)$,

(ii) $M_t \succeq L_t$,

(iii) $\phi L_t \succeq M_t$, and
(iv) $|M_t| = |L_t|$.

Proof. The proof is by induction on the number of steps. The hypothesis trivially holds at the beginning (for $t = 0$), when both sets M_0 and L_0 are empty and there are no active items.

For the induction step, suppose that $t > 0$ and that all claims (i)–(iv) hold right after time step $t' = t - 1$. Insertions and deletions of items at the beginning of step t do not affect the validity of claims of the lemma for sets L_{t-1}, M_{t-1} and Z_{t-1}, since they do not change these sets.

By the inductive assumption (iv) we have $|L_{t-1}| = |M_{t-1}|$, so if there is an item pending for MARK&PICK in step t, then there is also an unmarked item (not necessarily pending), which proves the first part of (i), i.e., the existence of m_t. To prove the second part of (i), i.e., that ℓ_t exists as well, we consider two cases. If m_t is not active, then, by the monotonicity, any item a pending for MARK&PICK satisfies $w(a) \geq w(m_t)$, so ℓ_t exists. Otherwise, when m_t is active, let B be the set of all items b such that $b \trianglerighteq m_t$. (So all items in B except m_t are in M_{t-1}.) By Fact 1, all items in B are active, and hence, by monotonicity, B contains the $|B|$ heaviest items inserted till step t. Since $m_t \in B \setminus M_{t-1}$, the assumption that $M_{t-1} \succeq L_{t-1}$ implies that L_{t-1} contains at most $|B| - 1$ items from B, so there is at least one item $a \in B \setminus L_{t-1}$. As $w(a) \geq w(m_t)$, this proves the existence of ℓ_t, completing the proof of (i).

In this step, MARK&PICK marks m_t and collects ℓ_t, so $M_t = M_{t-1} \cup m_t$ and $L_t = L_{t-1} \cup \ell_t$. We now apply Lemma 1 with Π being the set of all items inserted in steps 1, 2, ..., t, $V = M_{t-1}$, $v = m_t = \max(\Pi \setminus M_{t-1})$, $U = L_{t-1}$, and $u = \ell_t$. This implies that $M_t \succeq L_t$, showing (ii). Moreover, since $\phi L_{t-1} \succeq M_{t-1}$ and $\phi w(\ell_t) \geq w(m_t)$ by (i), we also obtain that $\phi L_t \geq M_t$, as claimed in (iii).

Lastly, since ℓ_t and m_t were chosen in step t, and $|L_{t-1}| = |M_{t-1}|$ by (iv), it follows that $|L_t| = |M_t|$ and so (iv) holds for step t, completing the inductive step and the proof of the lemma. □

Lemma 2 implies that Algorithm MARK&PICK is well-defined, that is, whenever MARK&PICK has a pending item then the items m_t and ℓ_t defined in the algorithm exist as well.

3.3. Partition into Phases

We now show that, without loss of generality, we can divide the computation into a sequence of disjoint and independent *phases*. In each phase the adversary collects items in each step, while MARK&PICK collects items up to a certain step and then stays idle till the end of this phase. The intuition behind the phase partition

is that if MARK&PICK does not have any pending items at some step then we can postpone all queue updates until the adversary completes collecting all his pending items.

We now formalize the definition of phase partitioning. Specifically, we want to show that, without loss of generality, we can assume that the input instance consists of a sequence of disjoint phases, where each phase satisfies the properties (a)–(e) below (where T denotes the number of steps of a phase and the steps in the phase are numbered $1, \ldots, T$):

(a) Both the adversary and MARK&PICK collect only items that were released in this phase.

(b) MARK&PICK marks only items that were released in this phase.

(c) MARK&PICK collects an item in each step $1, 2, \ldots, T-k$, for some k, and has no pending items in steps $T-k+1, \ldots, T$.

(d) The adversary collects an item in each step $1, \ldots, T$. Further, at step $T-k+1$, there are k active items not collected by the adversary, and the adversary collects these items in steps $T-k+1, \ldots, T$. (Thus right after step T all active items are collected by the adversary.)

(e) There are no insertions nor deletions in steps $T-k+1, \ldots, T$.

Note that we do not assume that all active items are deleted after step T; we explain the reason at the end of this section. The properties above are shown in the following lemma.

Lemma 3. Each instance I can be converted into an instance I' that has a partition into phases that satisfy properties (a)–(e). The gain of the adversary on I' is at least as large as that on I and the gain of MARK&PICK on I' is at most as large as that on I.

Proof. In the proof we will modify both the instance and the schedules of MARK&PICK and the adversary, so that the new schedules satisfy the conditions (a)–(e).

Our modifications of the adversary schedule may produce schedules that may not have the EEF property. Hence, for the purpose of our construction we will ignore the concept of the adversary forfeiting some items, and allow him to collect any item that is active but not yet collected. As explained earlier, the modified schedule can be converted into one that satisfies the EEF property, without decreasing the adversary's gain.

We first show the first property in (d): that it is possible to modify the instance and the adversary's strategy, so that he collects an item in each step. Note that we

can assume that the adversary always collects an item if there exists an active and not yet collected item, for otherwise the adversary can collect such an item now rather than later (or not at all). Thus, we only need to be concerned with steps where all active items have already been collected by the adversary.

Consider an arbitrary instance, and choose s such that MARK&PICK collects items in steps 1, 2, ..., s and does not have a pending item after step s. We can assume that $s \geq 1$, since otherwise no item was released at step 1 and we can simply remove this step from the instance.

Claim A. Without loss of generality, the adversary collects items in each step $1, 2, \ldots, s$.

To prove this claim, we show that if there is a step t, $1 \leq t \leq s$, where the adversary is idle, then we can modify the adversary strategy so that he collects all items collected before, plus one additional item; thus increasing his gain.

So suppose that such step t exists. We construct a sequence $t_0 = t, t_1, t_2, \ldots, t_q$ of different time steps such that $1 \leq t_i < t$ and $z_{t_i} = \ell_{t_{i-1}}$ for $i = 1, 2, \ldots, q$ and $\ell_{t_q} \notin Z_{t-1}$. This is quite straightforward: By our earlier argument, at step t all active items are already in Z_{t-1}; in particular $\ell_t \in Z_{t-1}$. It means that $\ell_t = z_{t_1}$ for some $1 \leq t_1 < t$. If $\ell_{t_1} \notin Z_{t-1}$ then $q = 1$. Otherwise, we find t_2 such that $z_{t_2} = \ell_{t_1}$, and so on. Since $|L_t| > |Z_{t-1}|$, such chain will eventually be found. We modify the adversary schedule by collecting each ℓ_{t_i} at step t_i, $i = 1, 2, \ldots, q$, and leaving other steps unchanged. Note that $\ell_{t_1}, \ldots, \ell_{t_q}$ are active in these steps, as MARK&PICK collects them at these steps. As a result, the adversary collects all items collected before, plus the item ℓ_{t_q} that was not collected in the previous schedule (because $\ell_{t_q} \notin Z_t$ and ℓ_{t_q} is not active at time t). The resulting schedule can be then converted into the EEF form.

This completes the proof of Claim A, and at this point we can assume that both MARK&PICK and the adversary collect items in steps $1, 2, \ldots, s$, and that MARK&PICK has no pending items right after step s.

Let k be the number of items that are active right after step s that have not been yet collected by the adversary.

Claim B. Without loss of generality, the instance contains steps $s+1$, ..., $s+k$. Furthermore,

(i) these steps do not involve any queue updates,

(ii) the adversary collects one item in each of these steps,

(iii) MARK&PICK is idle in these steps since it has no pending items.

Let q, $0 \leq q \leq k$, be the largest integer for which steps $s+1, \ldots, s+q$ satisfy the conditions in Claim B. Suppose that $q < k$. Then right after step $s+q$ there are still active items not collected by the adversary.

If step $s+q+1$ does not exist, add another step to the instance, with no queue updates, in which the adversary will collect another available item. Suppose that step $s+q+1$ exists. If there are queue updates in step $s+q+1$ (deletions or insertions), we do this: we "insert" a new step between $s+q$ and $s+q+1$ with no updates and have the adversary collect any active uncollected item. We then modify the subsequent collection schedule to take into account that this item is no longer available for the adversary after this step.

By iterating the above process, we will modify the computation so that Claim B is satisfied. We now define the first phase to consist of steps $1, 2, \ldots, T = s+k$. As needed, the adversary collects all items that were collected before, and perhaps collects more items, while MARK&PICK will mark and collect exactly the same set of items.

Iterating this procedure leads to a partition of all steps into subsequent phases. Properties (c)-(e) follow straight from the construction. To see that (a) holds as well notice that, according to the transformation, right after the last step of the phase, but before insertions in the next phase, both the adversary and MARK&PICK do not have any pending items. Thus, they cannot collect any item from this phase in the following phases.

Showing property (b) is a little more involved. One snag is that some items could be active during several phases, although only in the first of these phases they can be pending for the adversary or for MARK&PICK. First we show the following:

Claim C. When a phase ends, all active items are marked.

Consider the set A of the active items right after the current phase ends. By the definition of phases, they are all already collected by MARK&PICK, that is $A \subseteq L_\tau$, where τ is the total number of steps up to and including the current phase. By Fact 1, all deleted items are before each item in A in the queue order "$\trianglelefteq$". In other words, items of A form a suffix of all items released so far, i.e., they are the heaviest items released so far. Since also $L_\tau \preceq M_\tau$ (from Lemma 2), and $|L_\tau| = |M_\tau|$, we can conclude that $A \subseteq M_\tau$ as well. Thus Claim C holds.

Claim C gives us the following separation property: at the beginning of a phase, unmarked items from the previous phases are already deleted, and therefore they are lighter than the items inserted in the current phase. This allows us to show property (b): consider a t-th step of a phase, for $t \leq T-k$. At least t items were inserted in this phase till step t, and in step t at least one of them is not marked. As

mentioned above, this item is heavier than all unmarked items from the previous phases, so MARK&PICK will not mark any item from the previous phases. □

Properties (a)–(e) imply that there is no interaction between phases: in each phase items are marked and collected as if both MARK&PICK and the adversary started a new instance in this phase. For this reason, throughout the rest of the paper, we restrict our attention and the analysis to instances that consist of a single phase. Notice that there is a little subtlety here: it might happen that some items from previous phases are still active, though they cannot be marked nor collected by MARK&PICK or the adversary in this phase. Thus we can simply disregard these items in our analysis, as they do not affect the actions of the adversary and MARK&PICK; the computation in each phase would be identical if the previous phases did not exist at all. Hence, in the rest of the paper, whenever we write "active" or "deleted", we in fact mean "active and released in this phase" or "released in this phase and deleted", respectively.

A reader might ask at this point why we cannot simply delete all active items after a phase ends, thus making the phases completely disjoint. The reason is that this could violate the assumption about the weight monotonicity if items released in the next phase are lighter than the active items from the previous phase.

3.4. Competitive Analysis

As explained in the previous sub-section, we can assume from now on that the instance consists of one phase of T steps satisfying properties (a)–(e), with MARK&PICK collecting $T-k$ items in steps $1,2,...,T-k$, for some k, and the adversary collecting an item at each step.

Define $L'_t \subseteq L_t$ to be the set of items collected by MARK&PICK that are going to be collected by the adversary in the future, i.e.:

$$L'_t = L_t \cap (Z_T \setminus Z_t) \ .$$

We are particularly interested in the set L'_{T-k}.

Lemma 4. $L'_{T-k} = Z_T \setminus Z_{T-k} = \{z_{T-k+1}, \ldots, z_T\}$.

Proof. Since $L'_{T-k} = L_{T-k} \cap (Z_T \setminus Z_{T-k})$, it is enough to show that $Z_T \setminus Z_{T-k} \subseteq L_{T-k}$, i.e., that the items z_{T-k+1}, ..., z_T were collected by MARK&PICK before step $T-k$. This is quite straightforward: all these items are active in step $T-k+1$, because they are pending for the adversary. As MARK&PICK has no pending items at step $T-k+1$, it must have collected these items before step $T-k+1$. □

In some sense, the items in L'_{T-k} are "extra" for the adversary: in the last k steps, the adversary collects these k items while MARK&PICK is idle. A reader

may wonder whether L'_{T-k} simply contains the k heaviest items collected by MARK&PICK (in the current phase). However, this is not true: Fix any adversarial strategy for a single phase, and modify it by prepending it with k additional steps. At the beginning of these steps, the adversary releases a set H of k very heavy items that will be collected by both the adversary and MARK&PICK during initial k steps. The behavior of the algorithm and the adversary on the steps from the original strategy remains unchanged. This way, L'_{T-k} and H will be disjoint.

We would like to keep track of the steps that contribute extra items. Simply looking at the increases of $|L'_t|$ does not serve our purpose because this value may fluctuate, increasing and decreasing over time, while we need a quantity that increases monotonically. So for $t = 1, ..., T-k$ we define

$$\xi_t = \min_{t \le \tau \le T-k} |L'_\tau| \ .$$

By definition, $\xi_t \in \{\xi_{t-1}, \xi_{t-1}+1\}$ for all $t = 1, ..., T-k$ (we assume that $\xi_0 = 0$), and $\xi_{T-k} = k$. Now, the steps can be classified using the values of ξ_t. A step t is *increasing*, if $t \le T-k$ and $\xi_t = \xi_{t-1}+1$. All other steps are *non-increasing*; this includes steps $t > T-k$. Moreover, let

$$Z_t^\sharp = \{z_j \,:\, j \le t,\ j \text{ is an increasing step}\}, \text{ and}$$
$$Z_t^\circ = \{z_j \,:\, j \le t,\ j \text{ is a non-increasing step}\}.$$

Thus $|Z_t^\sharp| = \xi_t$ and $|Z_t^\circ| = t - \xi_t$ for $t \le T-k$.

We now describe the fundamental idea of the proof. We begin by presenting two simple instances, on which the competitive ratio of Algorithm MARK&PICK is not better than ϕ.

Example 1. Consider an instance in which all items are inserted at the beginning at once. There are $2n$ items in total, n items of weight 1 followed by n items of weight ϕ. (To simplify description, we allow items of equal weight in our examples.) After n steps all items are deleted. MARK&PICK will mark the n heavy items and collect the n light items, while the adversary can collect the n heavy items.

Example 2. The second instance is very similar: n items of weight 1 followed by n items of weight $\phi + \varepsilon$, for some small $\varepsilon > 0$, with all items inserted at the beginning. After the first n steps all items of weight 1 are deleted and the items of weight $\phi + \varepsilon$ are deleted after $2n$ steps. In the first n steps MARK&PICK will mark and collect the n heavy items, after which it will remain idle. The adversary can collect all items. The competitive ratio tends to ϕ with $\varepsilon \to 0$.

Let us now examine these two examples more closely and relate it to the notations introduced earlier. In Example 1, $T = n$ and the algorithm collects the same

number of items as the adversary, that is, $k = 0$. Furthermore, in each step L'_t is empty, hence all steps are non-increasing. Since $w(\ell_t) \geq w(m_t)/\phi$ at each step, MARK&PICK's gain is at least the weight of the marked items divided by ϕ, that is $\phi w(L_T) \geq w(M_T)$. On the other hand, no feasible schedule can have its gain larger than the total weight of the set of marked items (which may not even be feasible), whence the ϕ-competitiveness of MARK&PICK follows.

Example 2 represents the other extreme scenario, where the adversary collects twice as many items as MARK&PICK. Here we have $T = 2n$ and $k = n$. For all $t \leq T - k$ we have $L_t = L'_t$, and all the first n steps are increasing. It can be shown that for such instances $M_{T/2} = L_{T/2}$, i.e. MARK&PICK collects the $T/2$ heaviest available items. The idea here is that the items in $Z^{\sharp}_{T/2} = Z_{T/2}$ are light: in each step $t \leq T/2$ the item z_t was available for MARK&PICK but not chosen. Hence $w(z_t) \leq w(m_t)/\phi$. Summing over all $t \leq T/2$ gives $w(Z_{T/2}) \leq w(M_{T/2})/\phi$, the latter is equal to $L_{T/2}/\phi$, by our earlier observation. The items collected by the adversary in the "extra" steps are exactly the items collected by MARK&PICK, i.e. $M_{T/2}$. Thus, the total gain of the adversary is at most $w(Z_{T/2}) + w(M_{T/2}) \leq w(M_{T/2})/\phi + w(M_{T/2}) = \phi w(M_{T/2})$, as claimed.

In both instances MARK&PICK's gain is only $1/\phi$ times the gain of the adversary, although in the first instance MARK&PICK collects as many items as the adversary, whereas in the second one only half as many. The competitive analysis for these two types of instances requires different accounting: in the first case, each item a of the adversary is charged, one to one, to an item collected by MARK&PICK of weight at least w_a/ϕ; in the second case, we need to charge disjoint pairs of adversary's items, each to a different item of MARK&PICK, such that a pair a, b is charged to an item of weight at least $(w_a + w_b)/\phi$. An arbitrary instance could be a mixture of such two types of instances, in which case constructing such mappings becomes challenging. The rough idea of the analysis is to use the classification into non-increasing and increasing steps to define the two mappings. In a non-increasing step t, we would like to charge z_t to ℓ_t, since then $w(\ell_t) \geq w(z_t)/\phi$. For an increasing step t, we would like to charge z_t and ℓ_t (which will be collected by the adversary later) to $\ell_t = m_t$, since then $w(\ell_t) \geq (w(z_t) + w(m_t))/\phi$. But there are several difficulties that need to be overcome. For example, unlike in Example 2, in general it is not necessarily true that $m_t = \ell_t$ in increasing steps; as a matter of fact, m_t may not be collected by MARK&PICK at all, or it could be collected at some other step (increasing or non-increasing).

Although this intuition reflects the fundamental idea behind our argument, in the formal proof that follows we will not give an explicit construction of these two mappings; instead, we show how to relate the adversary gain and the algorithm's

gain to the weight of marked items. Precisely speaking, we will show that $M_{T-k} \succeq \phi Z_T^\sharp$ and $M_{T-k} \succeq Z_T^\circ$.

Lemma 5. If t is an increasing step, then
(i) $\ell_t \in Z_T \setminus Z_t$ (and thus $\ell_t \in L'_t$ as well),
(ii) $z_t \notin L_t$, and
(iii) $\phi w(z_t) < w(m_t)$.

Proof. Denoting $Z' = Z_T \setminus Z_t$, the definition of L'_t gives us that

$$L'_t \;=\; (L_{t-1} \cup \ell_t) \cap Z' \quad \text{and} \quad L'_{t-1} \;=\; L_{t-1} \cap (Z' \cup z_t) \;\; .$$

Thus, both these sets have cardinalities either $|L_{t-1} \cap Z'|$ or $|L_{t-1} \cap Z'| + 1$. Since t is an increasing step, $|L'_t| > |L'_{t-1}|$, and therefore $|L'_t| = |L_{t-1} \cap Z'| + 1$ and $|L'_{t-1}| = |L_{t-1} \cap Z'|$. The first equality implies that $\ell_t \in Z' = Z_T \setminus Z_t$, completing the proof of (i). The second equality implies that $z_t \notin L_{t-1}$. Since $\ell_t \notin Z_t$ and $z_t \in Z_t$, these two items are different, i.e., $\ell_t \neq z_t$. Therefore $z_t \notin L_{t-1} \cup \ell_t = L_t$, proving (ii).

Finally, we prove (iii). By (ii), $z_t \notin L_t$. Further, z_t must be active, because z_t is pending for the adversary at the beginning of step t. Therefore z_t is pending for MARK&PICK as well. By already proved (i), $\ell_t = z_{t'}$ for some $t' > t$. Both $z_{t'}$ and z_t are pending for the adversary at step t, yet the adversary collects z_t, which implies that $z_t \lhd z_{t'}$, by the EEF assumption. Also, both $z_{t'}$ and z_t are pending for MARK&PICK at the beginning of step t so, since MARK&PICK chooses $\ell_t = z_{t'}$, we can conclude that $\phi w(z_t) < w(m_t)$. The proof is now complete. □

Corollary 1. $M_{T-k} \succeq \phi Z_T^\sharp$.

Proof. Since steps $T-k+1, \ldots, T$ are non-increasing, $Z_T^\sharp = Z_{T-k}^\sharp$. We prove a more general claim, namely that $M_t \succeq \phi Z_t^\sharp$ for $t = 0, \ldots, T-k$.

The proof is by induction on t. The claim is clearly true for $t = 0$. Consider any step $t > 0$. If step t is non-increasing then $Z_t^\sharp = Z_{t-1}^\sharp$ and $M_t = M_{t-1} \cup m_t$, so the claim follows trivially. If step t is increasing, then by Lemma 5 (iii) and Lemma 1, $M_t = M_{t-1} \cup m_t \succeq \phi(Z_{t-1}^\sharp \cup z_t) = \phi Z_t^\sharp$. □

The innocent-looking lemma below is in fact the critical part of our analysis, as it will allow us to bound the adversary gain in non-increasing steps by the weight of marked items.

Lemma 6. The set Z_T° can be collected in the steps $1, \ldots, T-k$.

Proof. Note that the steps $T-k+1, \ldots, T$ are non-increasing by definition. Thus, by Lemma 4,

$$Z_T^\circ = Z_{T-k}^\circ \cup L'_{T-k} \;\; . \tag{1}$$

(The sum above is disjoint). We give a strategy of collecting items in steps 1, ..., $T-k$. By c_t we will denote the item collected in step t, and we let $C_t = \{c_1, c_2, \ldots, c_t\}$. For each t, set C_t will satisfy

$$Z_t^\circ \subseteq C_t \subseteq Z_t^\circ \cup L_t' \ . \tag{2}$$

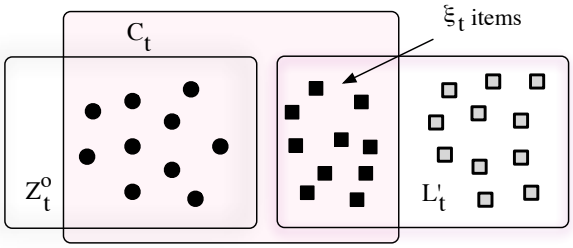

Figure 2: The relations between sets Z_t°, C_t and L_t'. Items in Z_t° are drawn as discs and those in L_t' as squares, with items in C_t filled solid.

Note that the sets Z_t° and L_t' are disjoint: the former is a subset of Z_t and the latter of $Z_T \setminus Z_t$. Further, $|Z_t^\circ| = t - \xi_t$ because exactly $t - \xi_t$ steps among 1, ..., t are non-increasing. This in turn implies that

$$|C_t \cap L_t'| = \xi_t \tag{3}$$

for all $t = 1, 2, ..., T-k$. See Figure 2 for illustration.

Next, let us check that condition (2) is sufficient to show the lemma. From (3) we have $|C_{T-k} \cap L_{T-k}'| = \xi_{T-k} = k$ and, by Lemma 4, $|L_{T-k}'| = k$. Thus, (2) and (1) imply that

$$C_{T-k} \ = \ Z_{T-k}^\circ \cup L_{T-k}' \ = \ Z_T^\circ \ ,$$

proving the lemma. It thus remains to show how the items in C_{T-k} can be collected.

The rest of the proof gives the strategy for collecting items c_1, ..., c_{T-k}. Consider a step t, and assume that condition (2) holds in steps 1, 2, ..., $t-1$. The choice of c_t depends on whether t is an increasing step and whether $z_t \in C_{t-1}$.

Case 1: t is an increasing step. By the case assumption,

$$Z_t^\circ \ = \ Z_{t-1}^\circ \ , \tag{4}$$

and moreover, by Lemma 5 (i),

$$L_t' \ = \ L_{t-1}' \cup \ell_t \ . \tag{5}$$

In this case, we will choose $c_t = \ell_t$. Trivially, $\ell_t \notin L'_{t-1}$ and, again by Lemma 5 (i), $\ell_t \notin Z_t$. Therefore the choice of c_t is valid, that is, $c_t \notin C_{t-1}$. So

$$C_t = C_{t-1} \cup \ell_t \ . \tag{6}$$

By (4), the induction assumption that (2) holds for $t-1$, and by (6), we have $Z_t^\circ = Z_{t-1}^\circ \subseteq C_{t-1} \subseteq C_t$. Similarly, using (6), the inductive assumption, (4), and (5), we get $C_t = C_{t-1} \cup \ell_t \subseteq Z_{t-1}^\circ \cup L'_{t-1} \cup \ell_t = Z_t^\circ \cup L'_t$. Thus (2) holds for step t.

Case 2: t is a non-increasing step. By the case assumption,

$$Z_t^\circ \ = \ Z_{t-1}^\circ \cup z_t \ . \tag{7}$$

Furthermore,

$$L'_{t-1} \ \subseteq \ L'_t \cup z_t \ . \tag{8}$$

We now distinguish two sub-cases. If $z_t \notin C_{t-1}$, then we we can choose $c_t = z_t$ obtaining

$$C_t \ = \ C_{t-1} \cup z_t \ . \tag{9}$$

Using (7), the inductive assumption, and (9), we have $Z_t^\circ = Z_{t-1}^\circ \cup z_t \subseteq C_{t-1} \cup z_t = C_t$. Similarly, from (9), the inductive assumption, (8), and (7), we have $C_t = C_{t-1} \cup z_t \subseteq Z_{t-1}^\circ \cup L'_{t-1} \cup z_t \subseteq Z_{t-1}^\circ \cup L'_t \cup z_t = Z_t^\circ \cup L'_t$. Therefore (2) holds for t in this sub-case.

The second sub-case is when $z_t \in C_{t-1}$. By the inductive assumption that (2) holds for step $t-1$, $Z_{t-1}^\circ \subseteq Z_{t-1}$, and $L'_{t-1} \cap Z_{t-1} = \emptyset$, we obtain that $C_{t-1} \setminus Z_{t-1} = C_{t-1} \cap L'_{t-1}$. Obviously, $z_t \notin Z_{t-1}$, and thus using (3),

$$|C_{t-1} \setminus Z_t| \ = \ |C_{t-1} \setminus Z_{t-1}| - 1 \ = \ \xi_{t-1} - 1 \ = \ \xi_t - 1 \ .$$

But $|L'_t| \geq \xi_t$, so $L'_t \setminus (C_{t-1} \setminus Z_t) \neq \emptyset$. This, in turn, implies that $L'_t \setminus C_{t-1} \neq \emptyset$, because $L'_t \cap Z_t = \emptyset$. We choose as c_t *any* item from $L'_t \setminus C_{t-1}$, and thus

$$C_t = C_{t-1} \cup c_t, \quad \text{for } c_t \in L'_t \ . \tag{10}$$

From (7), the inductive assumption, $z_t \in C_{t-1}$, and (10), we get $Z_t^\circ = Z_{t-1}^\circ \cup z_t \subseteq C_{t-1} \cup z_t = C_{t-1} \subseteq C_t$. To obtain the second inclusion in (2), we use (10), the inductive assumption, (8), (7), and the fact that $c_t \in L'_t$, getting $C_t = C_{t-1} \cup c_t \subseteq Z_{t-1}^\circ \cup L'_{t-1} \cup c_t \subseteq Z_{t-1}^\circ \cup L'_t \cup z_t \cup c_t = Z_t^\circ \cup L'_t$. This proves that (2) holds for step t, completing the proof. □

Corollary 2. $M_{T-k} \succeq Z_T^\circ$.

Proof. We prove the following, more general claim: if a set C of $T-k$ items can be collected in steps $1,2,...,T-k$ then $M_{T-k} \succeq C$. This will imply the lemma because in Lemma 6 we proved that the items in Z_T° can be collected in steps $1, \ldots, T-k$.

Fix a schedule for collecting the items in C. Let c_t be the item collected in step t and $C_t = \{c_1, c_2, \ldots, c_t\}$. So, in particular, $C = C_{T-k}$.

To prove our claim, we show that $M_t \succeq C_t$ after each step $t = 0, \ldots, T-k$. The proof is by induction on t. The claim trivially holds for $t = 0$. In any step $t > 0$ it is enough to apply Lemma 1 with $V = M_{t-1}$, $U = C_{t-1}$, $v = m_t = \max(\Pi \setminus M_{t-1})$, $u = c_t$, where Π is the set of items inserted in steps 1, 2, $\ldots$, t. □

Theorem 1. MARK&PICK is ϕ-competitive.

Proof. For the purpose of this proof let $\overline{m}_1, \ldots, \overline{m}_{T-k}$ denote the elements of M_{T-k} in order of decreasing weights. A similar convention is used for $L_{T-k} = L_T$. Recall that by the definition of a phase, monotonicity, and the marking scheme, $\overline{m}_1, \ldots, \overline{m}_k$ are the heaviest k items released in a phase. We first show that these items will be collected by MARKANDPICK, i.e., that

$$(\overline{m}_1, \ldots, \overline{m}_k) = (\overline{\ell}_1, \ldots, \overline{\ell}_k) \ . \tag{11}$$

By Lemma 2 (ii), $M_{T-k} \succeq L_{T-k} \succeq L'_{T-k}$. By Lemma 4, $|L'_{T-k}| = k$, and thus this dominance relation implies that $\{\overline{m}_1, \ldots, \overline{m}_k\} \succeq L'_{T-k}$. As all items in L'_{T-k} are active in step $T-k+1$, so must be the items $\overline{m}_1, \ldots, \overline{m}_k$, due to weight monotonicity and Fact 1. Since MARK&PICK has no pending item at this step, these items were already collected, which implies (11).

To prove the theorem, we will show that $w(Z_T) \leq \phi w(L_T)$. We now give a derivation of this inequality and later justify each step:

$$\begin{aligned}
w(Z_T) &= w(Z_T^\sharp) + w(Z_T^\circ) \\
&\leq \frac{1}{\phi} \sum_{i=1}^{k} w(\overline{m}_i) + \sum_{i=1}^{T-k} w(\overline{m}_i) && (12) \\
&= \phi \sum_{i=1}^{k} w(\overline{m}_i) + \sum_{i=k+1}^{T-k} w(\overline{m}_i) && (13) \\
&\leq \phi \sum_{i=1}^{k} w(\overline{\ell}_i) + \phi \sum_{i=k+1}^{T-k} w(\overline{\ell}_i) \ = \ \phi w(L_T) \ . && (14)
\end{aligned}$$

We now justify steps (12)-(14). By Corollary 2, $w(Z_T^\circ) \leq w(M_{T-k}) = \sum_{i=1}^{T-k} w(\overline{m}_i)$. Similarly, by Corollary 1 and $|Z_T^\sharp| = k$, it holds that $\{\overline{m}_1, \ldots, \overline{m}_k\} \succeq \phi Z_T^\sharp$, and hence $w(Z_T^\sharp) \leq \frac{1}{\varphi} \sum_{i=1}^{k} w(\overline{m}_i)$. Together, these bounds imply (12). Equation (13) follows from $1 + 1/\phi = \phi$ and simple algebra.

By (11), it holds that $\sum_{i=1}^{k} w(\overline{m}_i) = \sum_{i=1}^{k} w(\overline{\ell}_i)$, and Lemma 2 (iii) implies that $w(\overline{m}_i) \leq \phi w(\overline{\ell}_i)$ for each i. (Here we use the fact that there is a mapping supporting the dominance $\phi L_{T-k} \succeq M_{T-k}$ which maps $\overline{\ell}_i$ to $\overline{m}_i$, for each i.) Inequality (14) follows, and the proof is now complete. □

4. Conclusion

We have given a ϕ-competitive algorithm for collecting items with increasing weights from a dynamic queue, matching the lower bound for such instances. While attaining this competitive ratio for the general item collection problem is not possible [10], the best lower bound for a well-studied restriction of item collection, namely the bounded-delay packet scheduling problem, uses monotone instances [1, 3, 5]. Therefore, our results imply that if no ϕ-competitive packet scheduling algorithm exists, then the lower bound strategy used in the proof must be non-monotone. Additionally, if it is possible to achieve the ratio of ϕ for packet scheduling, the techniques introduced in this paper may be of help in designing a ϕ-competitive algorithm.

One interesting idea to pursue (as suggested by an anonymous referee) would be to study the dependence of the competitive ratio on the maximum number of items that can expire at any given step. If at most one packet is allowed to expire, then the algorithm that always collects the first item in the queue is 1-competitive. If more than one item is allowed to expire per step, the optimum ratio for monotone instances is ϕ (whether only two or more items can expire), as demonstrated by MARK&PICK and the trivial lower bound we have given in the Introduction. For general instances, the lower bound of ≈ 1.633 from [10] holds even if at most four items are allowed to expire at each step. Of course, the same question can be studied for the packet scheduling problem as well, where it is closely related to the study of so-called s-bounded instances [2, 14]. For s-bounded instances one can, without loss of generality, restrict attention to instances where at most s packets expire at each step.

Acknowledgements. We are grateful to the anonymous referees for numerous comments and suggestions regarding the presentation of the paper.

References

[1] N. Andelman, Y. Mansour, A. Zhu, Competitive queueing policies in QoS switches, in: Proc. 14th Symp. on Discrete Algorithms (SODA), ACM/SIAM, pp. 761–770.

[2] 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 (2006) 255–276.

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

[4] M. Englert, M. Westermann, Considering suppressed packets improves buffer management in QoS switches, in: Proc. 18th Symp. on Discrete Algorithms (SODA), ACM/SIAM, pp. 209–218.

[5] B. Hajek, On the competitiveness of online scheduling of unit-length packets with hard deadlines in slotted time, in: Proc. Conference on Information Sciences and Systems, pp. 434–438.

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

[7] A. Kesselman, Y. Mansour, R. van Stee, Improved competitive guarantees for QoS buffering, Algorithmica 43 (2005) 63–80.

[8] F. Li, J. Sethuraman, C. Stein, An optimal online algorithm for packet scheduling with agreeable deadlines, in: Proc. 16th Symp. on Discrete Algorithms (SODA), ACM/SIAM, pp. 801–802.

[9] F. Li, J. Sethuraman, C. Stein, Better online buffer management, in: Proc. 18th Symp. on Discrete Algorithms (SODA), ACM/SIAM, pp. 199–208.

[10] M. Bienkowski, M. Chrobak, C. Dürr, M. Hurand, A. Jeż, Ł. Jeż, G. Stachowiak, Collecting weighted items from a dynamic queue, 2012. To appear in Algorithmica, http://dx.doi.org/10.1007/s00453-011-9574-6.

[11] M. Chrobak, W. Jawor, J. Sgall, T. Tichý, Improved online algorithms for buffer management in QoS switches, ACM Trans. on Algorithms 3 (2007).

[12] W. Aiello, Y. Mansour, S. Rajagopolan, A. Rosén, Competitive queue policies for differentiated services, J. of Algorithms 55 (2005) 113–141.

[13] M. Bienkowski, M. Chrobak, C. Dürr, M. Hurand, A. Jeż, Ł. Jeż, G. Stachowiak, Collecting weighted items from a dynamic queue, in: Proc. 20th Symp. on Discrete Algorithms (SODA), ACM/SIAM, pp. 1126–1135.

[14] Ł. Jeż, One to Rule Them All: A General Randomized Algorithm for Buffer Management with Bounded Delay, in: Proc. 19th European Symp. on Algorithms (ESA), pp. 239–250.