Collecting Weighted Items from a Dynamic Queue*

Marcin Bienkowski† Marek Chrobak‡ Christoph Dürr§ Mathilde Hurand¶
Artur Jeż† Łukasz Jeż† Grzegorz Stachowiak†

Abstract

We consider the problem of collecting weighted items from a dynamic queue $\mathcal{S}$. Before each step, some items at the front of $\mathcal{S}$ can be deleted and some other items can be added to $\mathcal{S}$ at any place. An item, once deleted, cannot be re-inserted — in other words, it "expires". We are allowed to collect one item from $\mathcal{S}$ per step. Each item can be collected only once. The objective is to maximize the total weight of the collected items.

We study the online version of the dynamic queue problem. It is quite easy to see that the greedy algorithm that always collects the maximum-value item is 2-competitive, and that no deterministic online algorithm can be better than 1.618-competitive. We improve both bounds: We give a 1.89-competitive algorithm for general dynamic queues and we show a lower bound of 1.632 on the competitive ratio. We also provide other upper and lower bounds for restricted versions of this problem.

The dynamic queue problem is a generalization of the well-studied buffer management problem, and it is an abstraction of the buffer management problem for network links with intermittent access.

1 Introduction

We consider the problem of collecting weighted items from a dynamic queue $\mathcal{S}$ (an ordered list). Before each step, some items at the front of $\mathcal{S}$ (a prefix) can be deleted and some other items can be added to $\mathcal{S}$ at any place. An item, once deleted, cannot be re-inserted—in other words, it "expires". We are allowed to collect one item from $\mathcal{S}$ per step. Each item can be collected only once. The objective is to maximize the total weight of the collected items. We focus on the online version of this problem, where the updates of $\mathcal{S}$ are not known in advance. Our goal is to design online competitive algorithms and establish lower bounds.

To our knowledge, the problem above has not been explicitly addressed in the literature, though it naturally generalizes the well-studied problem of bounded-delay buffer management. In the buffer management problem, packets with weights and deadlines arrive in a buffer of a network link. (The weights represent various quality-of-service levels.) At each step, we can send one packet along the link. The objective is to maximize the total weight of packets sent before their deadlines. This is a special case of our dynamic queue problem, where packets are represented by items ordered according to deadlines. The difference is crucial though: in packet scheduling, packet arrival times are unknown but their deadlines are revealed at their arrival, while in dynamic queues *both* the arrival and deadlines are not known.

Competitive algorithms for various versions of bounded-delay buffer management problem have been extensively studied [3, 5, 6, 7, 8, 10, 11, 12, 13]. In particular, it is known that no deterministic online algorithm can have competitive ratio better than $\phi \approx 1.618$ [3, 6], and an algorithm with competitive ratio ≈ 1.828 has been recently developed [7] (see also [13]). Closing the gap between these bounds remains an intriguing open problem. For agreeable deadlines (where the items are released in order of non-decreasing deadlines), an upper bound of ϕ has been established [12].

The buffer management model above assumes that we can send one packet at each time step. This is not the case in networks when access to the link may be only intermittent. One example is that of a tiered QoS systems, where all traffic is divided into classes with different QoS guarantees. Packets from a lower tier are transmitted only if no higher-tier packets are pending. In this model, maximizing the total value of sent packets from this lower tier is equivalent to our dynamic queue problem. (A more explicit reduction will be given in the full version of this paper. The rough idea is that we can simulate item deletions by blocking time slots using tight top-tier packets,

*Supported by MNiSW grants number N206 001 31/0436, 2006–2008 and N N206 1723 33, 2007–2010, NSF grants OISE-0340752 and CCF-0729071, and ANR Alpage. Most of the work was carried out when M. Hurand was at LIX Ecole Polytechnique.

†Institute of Computer Science, University of Wrocław, 50-383 Wrocław, Poland. {mbi, aje, lje, gst}@ii.uni.wroc.pl

‡Department of Computer Science, University of California, Riverside, CA 92521, US.

§CNRS, LIX UMR 7161, Ecole Polytechnique, 91128 Palaiseau, France. durr@lix.polytechnique.fr

¶Google, 8002 Zürich Switzerland mathilde@google.com

that must be transmitted in the step when they are released.) Tiered systems are increasingly common—for example, the newly introduced WiMAX standard for the "last mile" connectivity comprises of five tiers with traffic ranging from the voice and video service with real-time guarantees to the lowest best-effort service for web-browsing and data transfer [2]. There are other scenarios where link access is intermittent or unpredictable, due to competition with other traffic streams, errors or interference in wireless channels, or link failures. (One extreme example is that of meteor burst communication, where a connection requires an entry of a meteor into the atmosphere at a desired location [1].)

The dynamic queue problem is also loosely related to various versions of online bipartite matching (it can be thought of as computing a maximum weight matching between time steps and items) and to the adwords problem, both extensively studied in the literature (see [9, 4, 14] and the references therein). Due to different focus and assumptions, however, algorithmic ideas developed for those problems do not seem to apply to dynamic queues.

Our results. It is quite easy to see that algorithm Greedy, which always collects the maximum-value item, is 2-competitive for general dynamic queues [8, 10]. We improve this bound, by providing a 1.89-competitive algorithm PrudentMark (see Section 3.) Our ratio is larger than the ratio of ≈ 1.828 for the more restricted problem of buffer management [7], but the algorithm in [7] (as well as the one in [13]) uses information about packet deadlines and is not applicable to dynamic queues. We also show that our analysis of PrudentMark is essentially tight.

Next, in Section 4, we study the special case of *FIFO queues*, where items can be added only at the end of the queue. The FIFO case generalizes the variant of buffer management with agreeable deadlines. For this case we give a 1.737-competitive algorithm EFH, and we show that the analysis of EFH is tight.

Our last upper bound is for the *non-decreasing weight case*, where the weights of the items are non-decreasing with respect to the list ordering. In Section 5 we present an online algorithm MarkAndPick for this case with competitive ratio ϕ—thus matching the lower bound from [3, 6]. This result has implications for buffer management, as the proof of the lower bound of ϕ uses instances with geometrically increasing weights. Thus, improving this lower bound—if possible at all—would require a completely new approach. Our competitive analysis uses an invariant technique involving dominance relations between sets of numbers, and is likely to find applications in improving upper bounds for various versions of buffer management problems.

We then turn our attention to lower bounds. It is easy to establish a lower bound of ϕ for any deterministic algorithm for dynamic queues, using only two items (see Section 6). We improve this bound, by proving a lower bound of ≈ 1.63. This bound applies even to the *decremental case*, where all items are inserted at the beginning and no insertions are allowed afterwards.

We also show two tight lower bounds for memoryless algorithms, which make decisions based only on the weights of the pending items. For deterministic algorithms we prove a lower bound of 2. This contrasts with a 1.893-competitive memoryless algorithm for buffer management [7]. Thus, for memoryless algorithms, knowing the exact deadlines helps. For randomized memoryless algorithms (against an adaptive adversary), we present a lower bound of $e/(e-1)$, matching an upper bound of Algorithm RMix [5], which can be adapted to dynamic queues.

2 Preliminaries

We refer to the items currently in $\mathcal{S}$ as *active*. In other words, those are the items that have been already inserted but not yet deleted. An item is called *pending* for an algorithm $\mathcal{A}$ if it is active but not yet collected by $\mathcal{A}$. We denote the weight of an item x as w_x and the total weight of a set of items X as $w(X)$. We use symbol "$\lhd$" to represent the ordering in $\mathcal{S}$, i.e. $a \lhd b$ means that a is before b in the list. This relation is well-defined, since items cannot be re-inserted into the list.

If all queue updates are specified upfront, an optimal solution can be computed in polynomial-time by reduction to maximum-weight matching: Represent the instance as a bipartite graph G whose partitions are items and time steps. (Only $O(n)$ time steps need to be considered, where n is the number of items.) An item a is connected to the time steps when a is active with edges of weight w_a. The maximum-weight matching in G represents an optimal collection sequence.

Using a routine exchange argument, it is easy to show that, without loss of generality, optimal (or the adversary's) solutions satisfy the following *Earliest-Expiration-First (EEF) Property:* If a, b are active at the same time, $a \lhd b$, and both a, b are collected, then a is collected before b. We say that a is *pending for the adversary* if it can be collected later by the adversary satisfying the EEF property. In other words, if at some step the adversary collects an item b, then he forfeits all active items $a \lhd b$, that is, they are no longer considered pending for the adversary.

An online algorithm $\mathcal{A}$ is called *R-competitive* if its gain on any instance I is at least the optimum gain on I divided by R. (An additive constant is sometimes allowed in this bound; in our upper bounds this constant is 0, and our lower bounds can be easily modified to work for this more general definition.) The *competitive ratio* of $\mathcal{A}$ is the smallest R for which $\mathcal{A}$ is R-competitive.

3 A 1.897-Competitive Algorithm for General Queues

It is easy to show that Algorithm Greedy, which always collects the maximum value item, is 2-competitive [8, 10]: Divide the items collected by the adversary into two types: (1) those collected by Greedy and (2) all other items. Obviously, the total weight of type-1 items does not exceed Greedy's gain. If the adversary collects a type-2 item x at time t, then x is also pending for Greedy at time t, so at this time Greedy collects an item at least as heavy as x. Thus the total weight of type-2 items also does not exceed Greedy's gain.

We improve this bound, by presenting an online algorithm PrudentMark with competitive ratio ≈ 1.897. (In the appendix, we show that our analysis is nearly tight, i.e. that the competitive ratio of PrudentMark is at least 1.894.) For simplicity, we assume that there are always pending items, for otherwise we can insert any number of items of zero weight into the queue, without affecting the analysis.

Algorithm PrudentMark**:** Fix two parameters $0 < \alpha, \beta < 1$. The algorithm maintains marks on some pending items. At each step, we proceed as follows:

```
/* update queue */
let $h$ be the heaviest pending item
let $m$ be the heaviest unmarked pending item
if $w_m < \alpha w_h$ then
    collect $h$
else
    mark $m$
    collect the earliest pending item $e$ with $w_e \geq \beta w_m$
```

THEOREM 3.1. *There are constants α and β for which Algorithm* PrudentMark *is 1.897-competitive.*

Analysis of PrudentMark**.** We choose β to be the only root of $\beta^3 - 4\beta^2 + \beta + 1 = 0$ in the interval $[0, 1]$, i.e. $\beta \approx 0.7261$, and $\alpha = 2 - 1/\beta \approx 0.6228$. For these parameters we prove that PrudentMark is R-competitive, where $R = 1/\beta^2 \approx 1.8967$. The following inequalities are routine to verify:

$$2\alpha \leq \beta+1 \leq 1/\beta^2 \leq 2 \leq 1/\beta+\alpha \leq 2\beta+\alpha \tag{3.1}$$

Types of items. We now focus our attention on the adversary's pending items. (Recall the definition in Section 2.) Our first observation, that follows directly from the algorithm, is this:

(i1) If item b is pending for PrudentMark then $w_b \leq w_h$. If PrudentMark marks an item m in this step then also $w_b \leq w_m/\alpha$.

Suppose that b is an item pending for the adversary. We call b an *M-item* if b has been marked by PrudentMark, and a *C-item* if b has been collected by PrudentMark. If b is both an M-item and a C-item then we refer to it as a *CM-item*.

We group some items into *protection pairs*. Each protection pair (b, k) consists of a C-item b and an M-item k. Item b is called the *protected item* and item k is called the *protecting item*. We also say that *k protects b*. These items will satisfy the following conditions at every step:

(i2) $b \lhd k$ and $w_b \geq \beta w_k$.

(i3) k is either an M-item pending for PrudentMark or is itself a protected CM-item.

(i4) there is no other protection pair where b is a protected item or k is a protecting item.

Conditions (i3) and (i4) imply that protection pairs form chains, where each item is protected by the next one. All items in a chain except last are C-items, all items except possibly first are M-items, and the last item in the chain must be a pending M-item. In general, not all C-items and M-items will belong to protection pairs; there may also be a number of *unprotected* C-items and *non-protecting* M-items present.

Proof idea. The proof uses amortized analysis. When PrudentMark collects an item of weight x, we think about it as receiving an allowance of Rx units. When the adversary collects an item of weight y, we pay him y units. The goal is to show that we never run out of money.

The idea of the analysis is to "pre-pay" for some items that are still pending for the adversary. At some steps we give adversary credit for his pending items, while in other steps we use these credits to decrease his gain in the step. The sum of his actual gain and the credit change will be called the *amortized gain.* We prove that the adversary's amortized gain in each step is at most R times our gain in this step.

Intuitively, let b be an item collected by PrudentMark, pending for the adversary, and with weight much larger than the weight of the PrudentMark's pending items. Then b already has full credit on it, for when the adversary collects b PrudentMark cannot afford to pay for b with the items that it still can collect.

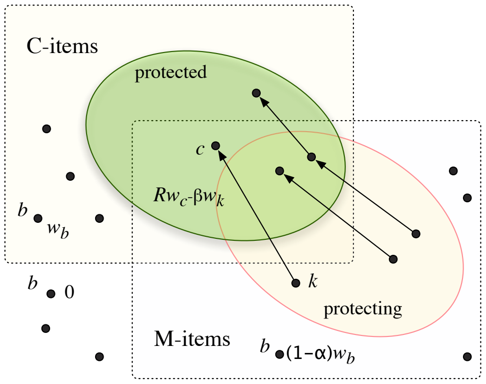

Figure 1: Items and their credits

In other situations, if b is not too large, we can afford to give the adversary only partial credit for b.

For example, if PrudentMark marks m and collects an item e pending for the adversary then (e, m) becomes a protection pair. If, at the same time, the adversary collects an item $z \lhd e$ that is pending for PrudentMark then we give the adversary only partial credit for e. The principle is that, for a fixed w_e, the smaller the value of w_m, the bigger the credit we give to e. This is because our "allowance" is Rw_e, while our "payment" is at most w_z, which cannot exceed βw_m (for otherwise PrudentMark would collect z instead of e). The intuition here is this: if m is very light, then e will actually receive full credit. On the other hand, if m is heavy, then we can afford to give e only partial credit, for if the adversary later collects e or m, then PrudentMark will also collect a relatively heavy item (since m is still pending for PrudentMark). We emphasize that the classification of e and m can change over time; for example e could become a CM-item or m could become a non-protecting M-item.

Assignment of credits. At a given step, let h and m be as described in PrudentMark, namely the heaviest pending item and the heaviest unmarked pending item, respectively. We present the strategy for awarding the adversary credits. If b is an item pending for the adversary, then at each step b satisfies:

(c1) If b is pending for PrudentMark and not marked then b has no credit.

(c2) If b is a non-protecting M-item pending for PrudentMark, it has a credit $(1 - \alpha)w_b$.

(c3) If b is an unprotected C-item then b has credit w_b.

(c4) Otherwise, b belongs to a protection pair. A protection pair (c, k) has credit $Rw_c - \beta w_k$. We split this credit between c and k as follows: k has credit $(1 - \alpha)w_k$, and c has credit $Rw_c - (1 + \beta - \alpha)w_k$. (By (i2) $Rw_c - (1 + \beta - \alpha)w_k \geq R\beta w_k - (1+\beta-\alpha)w_k = (1-\beta)w_k \geq 0$, so c's credit is non-negative.)

Amortized analysis. Fix one step, and let h and m be as described in PrudentMark. Let z be the item collected by the adversary in this step. We divide each step into two sub-steps: (i) In the first sub-step the adversary chooses z, inserts some items, and possibly deletes some items before z. In our analysis, this can cause some items other than z to change their status and we may need to update their credits. The total change of credits in this sub-step will be called the *adversary z-gain*. (ii) In the second sub-step, PrudentMark executes its move, collecting an item. This may result in some items changing their status, and thus we may have to adjust their credits. We refer to the credit change in this step as the *new credit*.

We start by showing bounds on the adversary z-gain.

FACT 3.1. (adv1) *If z is pending for* PrudentMark *then the z-gain is at most w_z.*

(adv2) *If z is a pending M-item then the z-gain is at most αw_z.*

(adv3) *If* PrudentMark *collects h in this step and z is neither an M-item nor a C-item (that is, z is pending for* PrudentMark *and not marked) then the z-gain is at most αw_h.*

(adv4) *Suppose that z is a C-item. If z is unprotected then the z-gain is at most 0. Otherwise, if z's protector is k, then the z-gain is at most $(1+\beta-\alpha)w_k - (R-1)w_z$. In either case, the z-gain can be bounded by any of the following quantities: $\alpha\beta w_k$, αw_z, $\alpha\beta w_h$, and, if* PrudentMark *marks m in this step, also by βw_m.*

Proof. First, since all credits are non-negative and only the items pending for the adversary have credits, if $a \lhd z$ then we will not increase the credit of a.

In cases (adv1), (adv2) and (adv3), we will not change the credit for any item a with $z \lhd a$. With this in mind, (adv1) is obvious, and so is (adv2), with the only difference being that in this case z already had credit $(1 - \alpha)w_z$. In (adv3), since PrudentMark collects h, we must have $w_z < \alpha w_h$, for otherwise PrudentMark could mark z. This implies (adv3).

It remains to show (adv4). Suppose that z was already collected by PrudentMark. If z is unprotected, then it already has full credit. Also, in this case we do not change the credit for any items a with $z \lhd a$, so the z-gain is at most 0.

Otherwise, suppose that z is protected by k. Then the adversary gain for collecting z is $w_z - [Rw_z - (1 +$

$\beta-\alpha)w_k] = (1+\beta-\alpha)w_k-(R-1)w_z$. Since $w_z \geq \beta w_k$, we can bound the last expression by $(1+\beta-\alpha)w_k - (R-1)w_z \leq (1+\beta-\alpha-(R-1)\beta)w_k = \alpha\beta w_k$.

If $w_k \leq w_h$, then we do not change credits on any items a with $z \lhd a$. Thus the z-gain is at most $\alpha\beta w_k$, and the remaining bounds follow from $w_k \leq w_z/\beta$, $w_k \leq w_h$ and $w_h \leq w_m/\alpha$.

Suppose now that $w_k > w_h$. Then w_k cannot be a pending M-item (by the definition of h), so w_k is a protected CM-item, by (i3). Let s be maximum for which there are protection pairs $(z = k_0, k = k_1)$, (k_1, k_2), $\ldots$, (k_{s-1}, k_s). Then k_s must be a pending M-item, so, by the definition of h, $w_{k_s} \leq w_h$.

To get the desired bound, we change the status and credits of some items: items $k_1, k_2, \ldots, k_{s-1}$ become unprotected C-items (with full credit). Item k_s becomes a pending, non-protecting M-item with credit $(1-\alpha)k_s$. This modification preserves the invariants and produces a correct credit assignment. Using (3.1), $w_{k_s} \leq w_h$ and $w_z \geq \beta w_k$, the z-gain is at most

$$\begin{aligned}
&\sum_{i=0}^{s-1} w_{k_i} + (1-\alpha)w_{k_s} - \sum_{i=0}^{s-1}\left[Rw_{k_i} - \beta w_{k_{i+1}}\right] \\
&= -(R-1)w_z + (1+\beta-\alpha)w_{k_s} - \sum_{i=1}^{s-1}(R-1-\beta)w_{k_i} \\
&\leq (1+\beta-\alpha)w_{k_s} - (R-1)w_z \\
&\leq (1+\beta-\alpha)w_h - (R-1)\beta w_h \\
&= \alpha\beta w_h \ .
\end{aligned}$$

Since, in this case, $w_h \leq w_k$, the z-gain is also bounded by $\alpha\beta w_k$, and the other bounds follow like in the previous case. $\square$

We show that, in this step, the invariant is preserved and that the adversary's amortized gain is at most R times the gain of PrudentMark. This is sufficient to prove R-competitiveness, by a standard amortization argument. The proof is by analyzing a number of cases.

Case 1: $w_m \geq \alpha w_h$. We mark m and collect the earliest pending item e such that $w_e \geq \beta w_m$.

Case 1.1: $z \lhd e$. We claim that the z-gain is at most βw_m. Indeed, if z is pending for PrudentMark then $w_z < \beta w_m$, thus, by (adv1), the z-gain is at most βw_m. If z is a C-item, then, by (adv4), the z-gain is at most βw_m as well.

We estimate the new credits. We have two sub-cases. If $e = m$ then e becomes an unprotected C-item with full credit w_m. Adding the βw_m z-gain, we get that the amortized adversary gain is at most $(1+\beta)w_e \leq Rw_e$.

Otherwise, suppose that $e \lhd m$. We create a protection pair (e, m), with total credit $Rw_e - \beta w_m$. Adding the βw_m bound on the z-gain, the adversary amortized gain is at most Rw_e. Note that if e was already a protecting M-item, then e becomes a protected CM-item, preserving (i3). Also, by PrudentMark, $w_e \geq \beta w_m$ so (i2) is preserved.

Case 1.2: $e \unlhd z$. We estimate the adversary z-gain. Suppose first that z is pending for PrudentMark. From (adv1), if $w_z \leq w_m$, the z-gain is at most $w_z \leq w_m$. If $w_z > w_m$, then z must be marked and $w_z \leq w_m/\alpha$, so, by (adv2), the z-gain is at most $\alpha w_z \leq w_m$. If z is a C-item, by (adv4), the z-gain is at most βw_m. We conclude that in this case the z-gain is at most w_m.

Next, we estimate the new credit. Since $e \unlhd z$, we do not need to give the adversary any credit for e. If $z \lhd m$, we give the adversary $(1-\alpha)w_m$ credit for m (which becomes a non-protecting M-item), and otherwise the credit increase is 0.

Overall, the adversary's amortized gain is at most $w_m + (1-\alpha)w_m = (2-\alpha)w_m \leq (2-\alpha)w_e/\beta = Rw_e$.

Case 2: $w_m < \alpha w_h$. In this case PrudentMark collects h. Note that h is a pending M-item. We first estimate the z-gain. If z is a pending M-item then the adversary gain is at most $\alpha w_z \leq \alpha w_h$, by (adv2). If z is pending for PrudentMark and not marked, then, by (adv3), the z-gain is at most $w_z \leq \alpha w_h$. If z is a C-item then, by (adv4), the z-gain is at most $\alpha\beta w_h \leq \alpha w_h$. We conclude that in this case the z-gain is at most αw_h.

We now have two sub-cases.

Case 2.1: $h = b_0$ protects some item b_1 and $z \lhd b_1$. Let s be the maximum index for which $(b_s, b_{s-1}), \ldots, (b_2, b_1), (b_1, b_0 = h)$ are protection pairs. Let also p be the maximum index for which $z \lhd b_p$. (Since $z \lhd b_1$, p is well-defined and $p \geq 1$.) All items $b_p, b_{p-1}, \ldots, b_0$ will become unprotected C-items. Using (3.1) and (i2), the total credit increase will be at most

$$\begin{aligned}
&\sum_{i=0}^{p} w_{b_i} - \sum_{i=1}^{p}\left[Rw_{b_i} - \beta w_{b_{i-1}}\right] \\
&= (1+\beta)w_h - (R-1-\beta)\sum_{i=1}^{p-1} w_{b_i} - (R-1)w_{b_p} \\
&\leq (1+\beta)w_h - (R-1-\beta)\sum_{i=1}^{p-1}\beta^i w_h - (R-1)\beta^p w_h \\
&= \left(\frac{1+\beta-\beta R}{1-\beta} - \frac{\beta^{p+1}(2-R)}{1-\beta}\right) w_h \\
&\leq \frac{1+\beta-\beta R}{1-\beta} w_h \ .
\end{aligned}$$

Adding the αw_h bound on the z-gain, substituting $\alpha = 2 - 1/\beta$ and $R = 1/\beta^2$, and using the definition of β, we conclude that the amortized adversary gain in this case is at most

$$\left(\alpha + \frac{1+\beta-\beta R}{1-\beta}\right) w_h = \frac{-\beta^2+4\beta-2}{\beta(1-\beta)} \cdot w_h = \frac{1}{\beta^2} \cdot w_h = R w_h \ .$$

Case 2.2: Either h is a non-protecting M-item, or it protects a C-item b such that $b \trianglelefteq z$. If $z \lhd h$, item h becomes an unprotected C-item with full credit, otherwise we do not change its credit. If h protects b, the case assumption implies that we do not give the adversary credit for b. Thus the increase of the credit will be at most αw_h. Adding the αw_h bound on the z-gain and using (3.1), we conclude that the amortized adversary gain is at most $2\alpha w_h \leq R w_h$.

4 A 1.737-Competitive Algorithm for FIFO Queues

Algorithm EFH: The computation is divided into stages, where each stage is a single step, a pair of consecutive steps, or a triple of consecutive steps. Formally, each stage begins in step (E) and ends right before the next step (E). We number the stages in the natural manner. EFH uses parameters, $\alpha, \beta, \xi \in [0,1]$, $\beta \leq \xi$, which we fix later.

```
(S) /* update queue */
(E) let $h$ be the heaviest pending item
    collect the earliest pending item $e$ with $w_e \geq \beta w_h$
    /* update queue */
(F) let $h'$ be the heaviest pending item
    if $h$ is not pending or $\alpha w_{h'} > w_h$ then goto (E)
    collect the earliest pending item $f$ with $w_f \geq \xi w_h$
    /* update queue */
(H) let $h''$ be the heaviest pending item
    if $h$ is not pending or $\alpha w_{h''} > w_h$ then goto (E)
    collect $h$ and goto (S)
```

THEOREM 4.1. *The competitive ratio of Algorithm* EFH *is $R = 2(\sqrt{13}-1)/3 \approx 1.737$ for constants $\beta = (\sqrt{13}+1)/8 \approx 0.576$, $\xi = \frac{4}{3}\beta = (\sqrt{13}+1)/6 \approx 0.768$, and $\alpha = \frac{3}{4} = 0.75$.*

Algorithm Analysis. We fix an instance and we compare EFH's gain on this instance to the adversary's gain. By the EEF property if the adversary collects items a, b, c in a stage, in this order, then $a \lhd b \lhd c$.

The proof is by amortized analysis. We associate credits with items that are pending for the adversary and collected by EFH. When EFH collects an item of weight w, it uses its allowance of Rw to pay for the item collected by the adversary. If there are any funds left, they can be assigned to some items as credits. If the allowance is not sufficient to pay for the adversary's item, we use credits to cover the difference.

We use the FIFO queue property in the following way. After nominating h in the stage we are guaranteed that no new items appear before h. Thus all the items collected by the adversary in this stage that are before h were either pending for EFH at (E) or had credit on them. On the other hand, all new items, including h' and h'', are after h, and thus if the adversary collects them then he gains no credit for items collected by EFH.

The adversary's amortized gain in a stage is the sum of weights of the items he collected in this stage plus the total credit change in the stage. We preserve the following invariant: after each stage, each item x pending for the adversary but already collected by EFH has credit of value w_x associated with it, with a sole exception: If e is the item collected by EFH in the last stage, then e can be designated as a *special item* with partial credit of $\frac{2}{3}w_e$, conditioned that the item h from the last stage is still pending (in other words, the last stage ended because $\alpha w_{h'} > w_h$ or $\alpha w_{h''} > w_h$).

LEMMA 4.1. *In each stage the adversary's amortized gain is at most R times* EFH*'s gain.*

Proof. We assume that, with the exception for the special item, at each step the adversary collects items that are not collected by EFH before this stage. Otherwise the adversary collects an item that has credit on it and his amortized gain for this item is zero.

We examine three disjoint cases, where Case $k = 1,2,3$ corresponds to a stage that lasts k steps. Within each case, we examine the reason for the stage to end (h' or h'' are very heavy, h was collected, or h was deleted) and the possible behavior of the adversary—the position of his $\rhd$-maximal item collected in this stage relatively to the items collected by EFH. In each of those subcases we show that the amortized gain of the adversary does not exceed R times the gain of EFH. The complete analysis will appear in the full version of the paper. □

5 A ϕ-Competitive Algorithm for Non-Decreasing Weights

In this section we give an online algorithm MarkAndPick that is ϕ-competitive for dynamic queues when item weights are increasing. More precisely, if $\lhd$ denotes the ordering of the items in the queue, then we assume that, at any time, for any two active items $a, b \in \mathcal{S}$, if $a \lhd b$ then $w_a \leq w_b$.

Algorithm MarkAndPick:

```
(U) /* update queue */
  if there is no pending item then skip step, go to (U)
```

```
let $h$ be the heaviest unmarked item (not necessarily active)
mark $h$, collect the earliest pending item $i$
with $w_i \geq w_h/\phi$
```

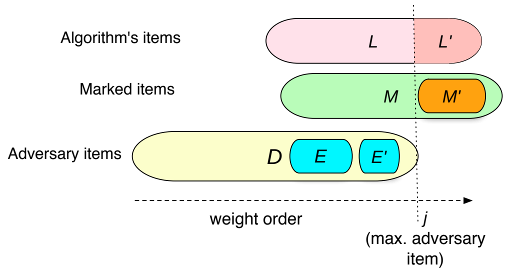

Figure 2: Notation.

Notation: The notation introduced here and used in the analysis is depicted on Figure 2. Let D_t, M_t, and L_t denote, respectively, the sets of items collected by the adversary, marked by the MarkAndPick, and collected by MarkAndPick up to and including step t. Let C_t be the set of items pending for the adversary at the end of step t. $L'_t = L_t \cap C_t$ is the set of items collected by MarkAndPick and pending for the adversary, and let $\ell_t = |L'_t|$. Similarly $e_t = |D_t| - |M_t|$ denotes the difference of the number of items collected by the adversary and by MarkAndPick up to time t. We also introduce the sets $E_t \subset D_t$, $E'_t \subset D_t$, and $M'_t \subset M_t$, such that $|E_t| = e_t$ and $|E'_t| = |M'_t| = \ell_t$. Intuitively, $M'_t = M_t \cup C_t$ and $E_t \cup E'_t$ represent *extra items* of the adversary: each time MarkAndPick picks an item i while the adversary picks e such that $e \lhd i$, then later the adversary might collect i and MarkAndPick has nothing to collect. Thus e is an "extra item" as it gives the adversary an "extra step". Items in E_t are extra items that are already "consumed"—the adversary used the extra step, while the ones in E'_t can still grant an extra step in the future.

Intuition: The ϕ-competitiveness comes from the combination of two extreme scenarios: In one, the adversary collects no extra items up to time ω. Then $|M_\omega| = |D_\omega| = |L_\omega|$, and whenever MarkAndPick marks an item h, it collects an item i with $w_i \geq w_h/\phi$. So its budget ϕw_i is sufficient to pay for h, and we get $\phi w(L_\omega) \geq w(M_\omega) \geq w(D_\omega)$. In the other scenario the adversary gets $|M_\omega|$ extra items. Then it can be shown that $L_\omega = M_\omega$, i.e. MarkAndPick collected the $|M_\omega|$ heaviest items. An item e is added to E_ω if MarkAndPick collects an item i after e even though e is pending; thus w_e is at most w_h/ϕ, where h was the item marked at that time. Overall, the gain of the adversary is $w(L_\omega)+w(E_\omega) \leq w(L_\omega)+w(L_\omega)/\phi = \phi w(L_\omega)$, and thus MarkAndPick has enough budget. To make the proof work for the intermediate cases, we need to introduce a dominance relations between sets and to represent our invariants in terms of this dominance relation.

Set dominance relation. Let X, Y be two finite sets of numbers. We say that X *dominates* Y, denoted $X \succeq Y$, if either $Y = \emptyset$, or $\max X \geq \max Y$ and $(X - \max X) \succeq (Y - \max Y)$. Note that we do not require that $|X| = |Y|$. In particular, $X \succeq \emptyset$, for any X.

For sets B and C of items, we say that B dominates C, writing $B \succeq C$, if $\{w_b : b \in B\}$ dominates $\{w_c : c \in C\}$. We write $B \succeq aC$ if $\{w_b : b \in B\}$ dominates $\{aw_c : c \in C\}$. The following lemma states the properties of the dominance relation needed in the analysis of MarkAndPick. Its proof appears in the full version of this paper.

Lemma 5.1. *Suppose that $X \succeq Y \neq \emptyset$. Then*

(i) $X - \min X \succeq Y - \min Y$. (ii) If $x \in X \cap Y$ then $X - x \succeq Y - x$. (iii) If $X, Y \subseteq Z$, $y \in Z - Y$, and $x \geq \max\{z \in Z - X : z \leq y\}$ then $X \cup x \succeq Y \cup y$. (In particular, this holds for $x \geq y$.)

Lemma 5.2. *For each time step t, there exist disjoint sets $E_t, E'_t \subseteq D_t$ with $|E_t| = e_t$ and $|E'_t| = \ell_t$, and a set $M'_t \subseteq M_t$ with $|M'_t| = \ell_t$, such that*

(a) $\phi w(L_t - L'_t) \geq w(M_t - M'_t) + w(E_t)$,

(b) $\phi L'_t \succeq M'_t \succeq L'_t$,

(c) $M_t \succeq (D_t - E_t - E'_t) \cup L'_t$, and

(d) $M'_t \succeq \phi E'_t$.

Proof. The idea is as follows: let j be the $\lhd$-maximal item collected by the adversary. We deal with items before and after j separately. We upper-bound the weight of items collected by the adversary before j by $w(M - M')$, so the maximum weight items among them, plus $w(E)$, thus extra items that were already used. Those items are paid by MarkAndPick items before j, that is $L - L'$, hence (a).

Among items after j the adversary will collect all the M' items. Then we are ready for two extreme scenarios of MarkAndPick choices. Either MarkAndPick collects light items and thus adversary's gain is paid by L', hence (b), or there are no items to collect by MarkAndPick and the adversary uses E' as extra items but cannot prevent MarkAndPick from taking the whole M', thus M' pays for M' and for E', which is formalized in (d).

Since $M' \succeq L'$, (c) should be viewed as $M - M' \succeq D - E - E'$, i.e. the adversary's non-extra items are dominated by marked items.

In the following, we turn this idea into a solid proof.

We show that the invariant in the lemma is preserved. For simplicity, we omit the subscript t and write $D = D_t$, $M = M_t$, etc. Also, let $\Delta w(D) = w(D_{t+1}) - w(D_t)$, $\Delta w(M) = w(M_{t+1}) - w(M_t)$, and so on. We view the process as follows: at each step, (I) the adversary first inserts items into $\mathcal{S}$; (II) then he selects the item j to be collected; (III) next, the adversary deletes some items from $\mathcal{S}$ (of course, only the items that are before j in $\mathcal{S}$ can be deleted); (IV) finally, both the adversary and the algorithm collect their items.

In order to show (a), we need to show that

$$\begin{aligned} \phi\Delta w(L) + \Delta w(M') \\ \geq\ \phi\Delta w(L') + \Delta w(M) + \Delta w(E)\ . \end{aligned} \tag{5.2}$$

In addition, we need to show that (b), (c) and (d) are preserved.

We look at all sub-steps separately. (I) Insertions do not affect the invariants. (None of the sets M, D, L, L' changes, and we do not change sets M', E, and E'.) In (II), suppose the adversary selects j and j' was an item collected by the adversary in the previous step. There may have been some items i, $j' \lhd i \lhd j$, that were in L'. Since these items are now removed from C, they are also removed from L', and we need to update M' and E' so that they have the same cardinality as L', and in such a way that the invariants are preserved. Let $i \in L'$ be such an item with minimum weight, g the minimum-weight item in M' and e the minimum-weight item in E'. We remove i from L', g from M' and e from E'. Since $\phi w_i \geq w_g$, by (b), we have $\phi\Delta w(L) + \Delta w(M') = 0 - w_g \geq -\phi w_i + 0 + 0 = \phi\Delta w(L') + \Delta w(M) + \Delta w(E)$, so (a) is preserved. Invariants (b) and (d) are preserved because we remove the minimum items from L', M', and E'. In (c), the left-hand side does not change and on the right-hand side we remove i from L' and add e to $D - E - E'$, and by (b) and (d) we have $w_i \geq w_g/\phi \geq w_e$, so the right-hand side cannot increase. In sub-step (III), deletions do not affect the invariants.

The rest of the proof is devoted to sub-step (IV). We examine (5.2) and the changes in (b), (c) and (d) due to the algorithm and the adversary collecting their items.

<u>Case A</u>: There is at least one pending item. The algorithm marks h and collects the earliest pending item i such that $w_i \geq w_h/\phi$. Thus h is added to M and i is added to L. We do not change E. We have some sub-cases.

<u>Case A.1</u>: $i \unlhd j$. Then i is not added to L', and we do not change E'. Since $\phi\Delta w(L) = \phi w_i \geq w_h = \Delta w(M)$ and $\Delta w(E) = 0$, to prove (5.2) it is now sufficient to show that

$$\Delta w(M') \quad\geq\quad \phi\Delta w(L'). \tag{5.3}$$

If $j \notin L$ (note that this included the case $i = j$), then L' does not change and we do not change M', so (5.3) is trivial. In (b) and (d) nothing changes. We add the maximum unmarked item h to M and j to D, so (c) follows from Lemma 5.1(iii).

If $i \neq j$ and $j \in L$ and then let g be the minimum item in M' and e the minimum item in E'. Item j is removed from L' and we remove g from M' and e from E'. Since, by (b), $\phi w_j \geq w_g$, we have $\Delta w(M') = -w_g \geq -\phi w_j = \phi\Delta w(L')$, so (5.3) holds. Since j, g and e are minimal, (b) and (d) are preserved. In (c), moving j from L' to D does not change the right-hand side. We also add h to M and e to $D - E - E' \cup L'$, so (c) is preserved because of the choice of h and Lemma 5.1(iii).

<u>Case A.2</u>: $i \rhd j$ and $j \notin L$. Then i is added to L'. We also add j to E'. Since $\Delta w(L) = \Delta w(L') = w_i$ and $\Delta w(M) = w_h$, to show (5.2) it is sufficient to show that

$$\Delta w(M') \quad\geq\quad w_h. \tag{5.4}$$

Since $|L'|$ increased, we need to add one item f to M'. We choose this f as follows: If $i \unlhd h$, we choose $f = h$. Otherwise, if $i \rhd h$ then, by the choice of h, we get that all items $h \unlhd f \unlhd i$ are now marked. In this case, we choose the largest $f \unlhd i$ such that $f \notin M'$ and add it to M'. Since $h \in M - M'$, h itself is a candidate for f, so we have $h \unlhd f \unlhd i$.

Note that, in this case, by the choice of i (as the earliest pending item with weight at least w_h/ϕ), $j \lhd i$ and $j \notin L$, we have $w_f \geq w_h \geq \phi w_j$. In particular, this means that $j \lhd h$ and that all items $h \unlhd f' \lhd i$ are in L'.

Since $w_f \geq w_h$, (5.4) is trivial. Invariant (d) is also quite easy, since $w_f \geq \phi w_j$, by the previous paragraph. In (c), adding j to D and E' does not change the right-hand side. We also add h to M and i to L', which preserves (c) by the choice of h and Lemma 5.1(iii).

To show (b), if $i \unlhd h$ then $f = h$ and, since $\phi w_i \geq w_h \geq w_i$, invariant (b) is preserved. If $i \rhd h$, then, $\phi w_i \geq w_i \geq w_f$, so the first part of (b) is preserved. That the second part of (b) is preserved follows from the choice of f and Lemma 5.1(iii).

<u>Case A.3</u>: $i \rhd j$ and $j \in L$. Then we remove j from L' and add i. We thus have $\Delta w(L) = w_i$, $\Delta w(L') = w_i - w_j$ and $\Delta w(M) = w_h$. Thus to show (5.2) it is sufficient to show that

$$\Delta w(M') + \phi w_j \quad\geq\quad w_h. \tag{5.5}$$

We do not change E'. To update M', we proceed as follows. Let g be the lightest item in M'. Since j is the minimal element of L', (b) implies $w_j \le w_g \le \phi w_j$. We first remove g from M'. Next, we proceed similarly as in the previous case, looking for an item f that we can add to M' to compensate for removing g (since $|M'|$ cannot change in this case.) Let $h' = \max(g, h)$. Note that $h' \in M - M'$. If $i \trianglelefteq h'$, we choose $f = h'$. Otherwise, if $i \triangleright h'$ then, by the choice of h', we get that all active items $h' \trianglelefteq f \trianglelefteq i$ are marked. In this case, we choose the largest $f \trianglelefteq i$ such that $f \notin M'$ and add it to M'. Since $h' \in M - M'$, h' itself is a candidate for f, so we have $h' \trianglelefteq f \trianglelefteq i$.

Note that, in this case, by $j \trianglelefteq g$, all items $h' \trianglelefteq f \triangleleft i$ are active and, by the choice of i, they are all in L'.

Now, in (5.5) we have $\Delta w(M') + \phi w_j = (-w_g + w_f) + \phi w_j \ge w_f \ge w_{h'} \ge w_h$. In (d), the left-hand side can only increase (since $f \ge g$) and the right-hand side does not change. In (c), moving j from L' to D does not change the right-hand side. We also added h to the left-hand side and i to L' on the right-hand side, so (c) is preserved by the choice of h and Lemma 5.1(iii).

In (b), removing j from L' and g from M' does not affect the invariant. Then we add f to M' and i to L'. By the algorithm, we have $\phi w_i \ge w_h$, while by the case assumption and (b), we have $\phi w_i \ge \phi w_j \ge w_g$. Therefore $\phi w_i \ge w_{h'}$. Since either $f = h'$ or $f \trianglelefteq i$, this implies $\phi w_i \ge w_f$, showing that the first inequality in (b) is preserved. The second part of (b) follows again from the choice of f and Lemma 5.1(iii).

Case B: There are no pending items for the algorithm. It means that L' contains all active items $i \trianglerighteq j$, including j. By the weight ordering assumption and the second part of (b) this implies that $L' = M'$. Since the adversary collects an item and the algorithm does not, $e = |D| - |M|$ increases by 1, so we also need to add an item to E. Let b be the minimum-weight item in E'. We do this: we remove j from M' and from L' and we move b from E' to E. Using the choice of b and (d) we have $w_j \ge \phi w_b$, so $\phi \Delta w(L) + \Delta w(M') = 0 + (-w_j) = -\phi w_j + w_j/\phi \ge\ge -\phi w_j + 0 + w_b = \phi \Delta w(L') + \Delta w(M) + \Delta w(E)$, and thus (5.2) holds. By the choice of j and e as the minimum items in L' and E', respectively, invariants (b) and (d) are preserved. In (c), j moves from L' to D, and b moves from E' to E, so the right-hand side does not change. □

Theorem 5.1. *Algorithm* MarkAndPick *is ϕ-competitive for dynamic queues if item weights are non-decreasing.*

Proof. By the invariants of Lemma 5.2, at each time step it holds that $\phi w(L_t) \ge [\phi w(L'_t) - w(M'_t)] + w(M_t) + w(E_t) \ge 0 + [w(D_t - E_t - E'_t) + w(L'_t)] + w(E_t) = w(D_t) + w(L'_t) - w(E'_t) \ge w(D_t) + w(M'_t)/\phi - w(E'_t) \ge w(D_t)$, and the ϕ-competitiveness follows. □

6 Lower Bounds

As the item collection problem is a generalization of the buffer management problem, we immediately get a lower bound of ϕ [3, 6] for the competitive ratio of any deterministic algorithm. However, for our problem the proof can be substantially simplified: Start with two items, $a \triangleleft b$, with weights $w_a = 1$ and $w_b = \phi$. If the algorithm chooses b, the adversary chooses a, deletes it, and collects b in the next step, so the ratio is $(w_a + w_b)/w_b = (1+\phi)/\phi = \phi$. If the algorithm chooses a, the adversary chooses b and deletes both items, so the ratio is $w_b/w_a = \phi$ again.

Lower bound of 1.63. In the remainder of this section, we show how to improve the lower bound to 1.63. Our lower bound works even in the decremental case, however, for simplicity reasons, we show it first for the general case. The proof is by presenting an adversary's strategy that forces any deterministic online algorithm $\mathcal{A}$ to gain less than 1/1.63 times the adversary's gain.

Adversary's strategy. Note that since the algorithm is deterministic, the adversary knows the moves of the algorithm in advance. In particular, we can specify the items he collects at the end.

We assume that the items appear gradually, so that at each step the algorithm has at most three items to choose from. Fix some $n \ge 2$. To simplify notation, in this section we refer to items simply by their weight, thus below "z_i" denotes both an item and its weight. The instance consists of a sequence of $2n$ items $1, z_1, z_2, \ldots, z_{2n-2}, z_{2n}$ (note that no item is indexed by $2n-1$) such that

$$z_2 \triangleleft z_4 \triangleleft \ldots \triangleleft z_{2n-2} \triangleleft z_{2n} \triangleleft z_{2n-3} \triangleleft \ldots \triangleleft z_3 \triangleleft z_1 \triangleleft 1 \ ,$$
$$\text{and } 1 > z_1 > z_2 > \ldots > z_{2n-3} > z_{2n-2} > z_{2n} > 0 \ .$$

The even- and odd-numbered items in this sequence form two roughly geometric sequences. In fact, z_{2i} is only slightly smaller than z_{2i-1}, for all $i = 1, \ldots, n-1$.

Initially, items $z_2 \triangleleft z_1 \triangleleft 1$ are present. In step $i = 1, 2, \ldots, n-1$, the adversary maintains the invariant that the active items are $z_{2i} \triangleleft z_{2i-1} \triangleleft \ldots \triangleleft 1$, of which only three items z_{2i}, z_{2i-1} and 1 are pending for $\mathcal{A}$ (i.e. $\mathcal{A}$ already collected $z_{2i-3}, \ldots, z_1$). The adversary's move depends now on what $\mathcal{A}$ collects in this step:

(i) $\mathcal{A}$ collects z_{2i}. Then the adversary ends the game by deleting all active items. In this case the adversary collects i heaviest items: $1, z_1, z_2, z_3, \ldots, z_{i-1}$.

(ii) $\mathcal{A}$ collects 1. The adversary ends the game by deleting x_{2i} and x_{2i-1}. This leaves $\mathcal{A}$ with no pending items, and the adversary can now collect $\mathcal{A}$'s items one by one. Overall, in this case the adversary collects $2i$ heaviest items: $1, z_1, z_2, \ldots, z_{2i-2}, z_{2i-1}$.

(iii) $\mathcal{A}$ collects z_{2i-1}. In this case the game continues. If $i < n-1$, the adversary deletes z_{2i}, inserts z_{2i+2} and z_{2i+1} into the current list (according to the order defined earlier), and the game proceeds to step $i+1$. The case $i = n-1$ is slightly different: here the adversary only inserts the last item z_{2n} before proceeding to step n (described below).

If the game reaches step n, $\mathcal{A}$ has two pending items, z_{2n} and 1. In this step, the adversary behavior is similar to previous steps: if $\mathcal{A}$ collects z_{2n}, then the adversary deletes the whole sequence and collects n heaviest items: $1, z_1, z_2, z_3, \ldots, z_{n-1}$. If $\mathcal{A}$ collects 1, the adversary deletes z_n, leaving $\mathcal{A}$ without pending items, and allowing the adversary to collect the whole sequence.

Lemma 6.1. *Suppose that there is a sequence* $1, z_1, \ldots, z_{2n-2}, z_{2n}$, *and a constant* R, *such that for all* $0 \le j < n$ *it holds that*

$$R \cdot (1 + \textstyle\sum_{i=1}^{j} z_{2i-1}) \le 1 + \sum_{i=1}^{2j+1} z_i \ ,$$
$$R \cdot (z_{2j+2} + \textstyle\sum_{i=1}^{j} z_{2i-1}) \le 1 + \sum_{i=1}^{j} z_i \ .$$

Then there is no R-competitive deterministic online algorithm.

The lemma is clear from the description of the strategy given earlier, since the sides of the inequalities above represent the gains of the adversary and the algorithm in various steps. Additionally, the lemma holds even if all the items are inserted at the beginning. To achieve this, we slightly modify the adversary's strategy: all the items are present at the beginning, and whenever $\mathcal{A}$ deviates from the choices (i), (ii), (iii), it collects an item lighter than z_{2i}, and thus the adversary can finish the game as in Case (i). Lemma 6.1 and straightforward calculations for $n = 3$ (6 items) yield the following.

Theorem 6.1. *There is no deterministic online algorithm for dynamic queues (even for the decremental case) with competitive ratio smaller than* 1.6329.

A natural question arises how much this bound can be improved with sequences $\{z_i\}$ of arbitrary length. For $n = 5$ (10 items), one can obtain $R = 1.6367\ldots$, and our experiments indicate that the corresponding ratios tend to ≈ 1.6378458, so the improvement is minor.

Memoryless algorithms. Lower bounds can be improved for memoryless algorithms, i.e. algorithms that decide which item to collect based only on the weights of their pending items. The details will appear in the full version of the paper.

Theorem 6.2. *No deterministic memoryless algorithm has competitive ratio smaller than* 2.

Theorem 6.3. *No randomized memoryless algorithm has competitive ratio smaller than* $\frac{\mathrm{e}}{\mathrm{e}-1}$ *against an adaptive-online adversary.*

References

[1] Meteor burst communications. `http://en.wikipedia.org/wiki/Meteor_burst`.

[2] WiMAX. `http://en.wikipedia.org/wiki/WiMAX`.

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

[4] B. Birnbaum and C. Mathieu. On-line bipartite matching made simple. *SIGACT News*, 39(1):80–87, 2008.

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

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

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

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

[9] B. Kalyanasundaram and K. Pruhs. An optimal deterministic algorithm for online b-matching. Manuscript, 1996.

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

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

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

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

[14] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54:22, 2007.