Abstract

We provide a simplified proof of the 3-competitiveness of the greedy algorithm for scheduling weighted packets in the multi-FIFO buffer model. Azar and Richter provided a proof using the zero-one principle (Azar and Richter. STOC 2004). We use a different approach and we hope our approach can lead to an improved FIFO buffering algorithm.

1 Problem Setting

Time is discrete as time steps. Online algorithms have no knowledge about one packet’s characteristics until it actually arrives. There are \( m \geq 2 \) buffers called \( Q_1, Q_2, \ldots, Q_m \) respectively. At any time, the buffer \( Q_i \) can store at most \( b_i \in \mathbb{Z}^+ \) packets. All the buffers are preemptive. Packets arrive over time. Each packet \( p \) is associated with an integer release date \( r_p \in \mathbb{Z}^+ \), a non-negative value \( w_p \in \mathbb{R}^+ \), and one target buffer that it can reside in. Arriving packets may be buffered at their destined buffers for future delivery. In each time step, only one pending packet is allowed to be sent. The order of sent packets should comply with the order of their release dates (which is called the First-In-First-Out order). Our objective is to maximize the total value gained by delivering packets in an online manner. In this model, the online policy has to make the decision of selecting one buffer to send a packet, as well as managing individual buffers. Even the variant in which all packets have the same value is not a trivial problem.

2 Algorithm \( \text{MQ} \)

Azar and Richter propose a greedy algorithm called TLH on the multi-FIFO buffer model in [1]. We name it \( \text{MQ} \) (Multiple Queues) and briefly introduce the algorithm again here.

There are \( m \) buffers \( Q_1, Q_2, \ldots, Q_m \). For each buffer, we use the Greedy policy to accept and send packets. In each time step, \( Q_i \) accepts packets which are assigned to it greedily and sends the first packet whenever \( Q_i \) is allowed to send a packet. Consider an arbitrary time step \( t \). \( \text{MQ} \) works as in Algorithm 1.

Algorithm 1 \( \text{MQ} \)

1: Run Greedy on buffers \( Q_1, \ldots, Q_m \) respectively.
   (That is, drop the minimum-value packet if the buffer is full.) Append newly accepted packets, if any, at the end of the packet queues.
   The \( m \) candidate packets to send (which are the first packets of those \( m \) buffers) are \( e^1_i, \ldots, e^m_i \).
2: Send \( e^i_i \) with the max\{\( w_{e^1_i}, \ldots, w_{e^m_i} \)\}.

3 \( \text{MQ} \)'s Competitive Analysis

Azar and Richter [1] show that \( \text{MQ} \) is 3-competitive, based on an analysis including 0/1 principle and a marking scheme.

Theorem 1 Zero-one Principle. [1] Let \( \mathcal{A} \) be a comparison-based switching algorithm (deterministic or randomized). \( \mathcal{A} \) is a \( c \)-approximation algorithm if and
only if \( A \) achieves \( c \)-approximation for all packet sequences whose value are restricted to 0/1.

We will prove that MQ is 3-competitive using a different analysis from a marking scheme method. The basic idea is that we compare MQ with another overcharged online algorithm, and the over-charged value is regarded as the gain of our algorithm’s adversary ADV. MQ and ADV have identical buffers at the beginning of each time step.

We still include zero-one principle in our analysis of MQ. Note that MQ is a comparison-based algorithm. We consider MQ’s competitiveness for instances with \((0/1)\)-value packets only, and this competitiveness is the same as MQ runs over instances with arbitrary packet values.

Let \( OPT \) denote the optimal offline algorithm, let \( O \) be the set of packets sent by \( OPT \). Let \( ADV \) denote MQ’s adversary.

**Theorem 2** MQ is 3-competitive.

**Proof 1** Our proof works as follows. We create an online algorithm called ADV. We also illustrate a procedure of overcharging ADV. In order to prove Theorem 2, we should create ADV and define its charging scheme such that:

a. MQ and ADV have identical buffers at the beginning/end of each step’s delivery.

b. We overcharge ADV such that the total overcharged value is no less than what \( OPT \) gains, that is, the total value charged on ADV is \( \geq \sum_{j \in O} w_j \).

c. MQ gains at least \( 1/3 \) of the overcharged value of ADV in each time step.

We assume that ADV selects the same queue as OPT, but schedules the first packet that Greedy expects to send. Notice that in some time steps, ADV does not really send a packet (i.e., the first packet is still in ADV’s buffer); however, we still let ADV “gain” some value to favor ADV (e.g., we charge \( w_i \) to ADV).

Given a time step \( t \), assume the queue \( MQ \) selects is \( Q_i \), the packet \( MQ \) sends is \( e^t_i \). MQ’s gain is \( W_t := w_{e^t_i} \). Let \( Q_j \) be the queue \( ADV \) selects — (\( Q_j \) is also the queue OPT selects.) Let ADV’s overcharged value be \( V_t \).

1. Assume \( Q_i = Q_j \).

   \[
   W_t := w_{e^t_i} \quad \text{and we let } V_t := 2 \cdot w_{e^t_i} = 2 \cdot w_{e^t_i}.
   \]

   Notice that at the end of this step, MQ and ADV have identical buffers and they will have identical buffers after both applying the greedy admission of new packets for the next step. Therefore, above requirements [a.] and [c.] are satisfied.

   We will show that charging \( 2 \cdot w_{e^t_i} \) for ADV in this step satisfies the requirement [b.] in Lemma 1.

2. Assume \( Q_i \neq Q_j \).

   We do not remove \( e^t_i \) from \( Q_j \); but at first, we charge \( w_{e^t_i} \) for ADV. (If \( e^t_i \) is an overloaded packet with \( w_{e^t_i} = 1 \), we assume ADV “sends” the overloading packet but leaves \( e^t_i \), which becomes underloaded, in the buffer \( Q_j \).) We will show that charging \( w_{e^t_i} \) for ADV due to \( e^t_i \)’s existence in \( Q_j \) at the end of this step satisfies the requirement [b.] in Lemma 1.

   Also, we remove \( e^t_j \) from \( Q_i \) for MG. \( e^t_j \) should be in ADV’s buffer at the end of this step. To keep MG and ADV have identical buffers, it only favors ADV if we let ADV “send” one more packet \( e^t_j \). \( e^t_j \) may be an overloaded packet (i.e., an \( O \)-packet is not accepted by ADV due to \( e^t_i \) in ADV’s buffer) and thus, we charge \( 2 \cdot w_{e^t_i} \) for ADV as well in this step.

   All other queues are the same for MQ and ADV at the end of this step. In this way, we keep ADV and MQ have identical buffers at the end of this step’s delivery.

   The charged value ratio in this step is bounded by

   \[
   \frac{V_t}{W_t} = \frac{2 \cdot w_{e^t_i} + w_{e^t_i}}{w_{e^t_i}} \leq \frac{w_{e^t_i}}{w_{e^t_i}} + 2 \leq 3.
   \]

   ■

**Lemma 1** The charging scheme for ADV in the proof of Theorem 2 satisfies requirement [b.].

**Proof 2** First of all, due to 0/1 principles, the packets sent by OPT, i.e., \( O \), should be a subset of those packets with value 1.

Consider the Greedy policy in accepting packets (for both ADV and MG). At the time when a new arrived packet \( p \in O \) is rejected by ADV, the destined queue \( Q_i \) for \( p \) must be full of packets with value 1, and there must exist one packet \( q \) that has been sent by OPT but is still in ADV’s buffer. We can map \( p \) to the earliest such packet \( q \) in the buffer \( Q_i \). Thus, any packet \( q \) with value 1 in ADV’s buffer \( Q_i \) is overloaded by at most one unsent \( O \)-packet. All packets before such overloaded packet \( q \) (if any) are packets with value 1.

Notice that the packets in the input sequence are with \((0/1)\)-values only, and thus, if ADV sends a packet with value 1 in \( Q_i \), and if the first packet in \( Q_i \) is a packet with value 1, we can always regard ADV sends the first packet in this time step. Also, if the first packet in queue \( Q_i \) is one with value 0, any packet in \( Q_i \) is not overloaded (please refer to the charging scheme for ADV). Therefore, Lemma 1 holds. ■
References