# A $\phi$-Competitive Algorithm for Scheduling Packets with Deadlines

Pavel Veselý* Marek Chrobak† Łukasz Jeż‡ Jiří Sgall§

**Abstract**

In the *online packet scheduling problem with deadlines* (PacketScheduling, for short), the goal is to schedule transmissions of packets that arrive over time in a network switch and need to be sent across a link. Each packet has a deadline, representing its urgency, and a non-negative weight, that represents its priority. Only one packet can be transmitted in any time slot, so, if the system is overloaded, some packets will inevitably miss their deadlines and be dropped. In this scenario, the natural objective is to compute a transmission schedule that maximizes the total weight of packets which are successfully transmitted. The problem is inherently online, with the scheduling decisions made without the knowledge of future packet arrivals. The central problem concerning PacketScheduling, that has been a subject of intensive study since 2001, is to determine the optimal competitive ratio of online algorithms, namely the worst-case ratio between the optimum total weight of a schedule (computed by an offline algorithm) and the weight of a schedule computed by a (deterministic) online algorithm. We solve this open problem by presenting a $\phi$-competitive online algorithm for PacketScheduling (where $\phi \approx 1.618$ is the golden ratio), matching the previously established lower bound.

## 1 Introduction

In the *online packet scheduling problem with deadlines* (PacketScheduling, for short), the goal is to schedule transmissions of packets that arrive over time in a network switch and need to be sent across a link. Each packet $p$ has a deadline $d_p$, representing its urgency, and a non-negative weight $w_p$, that represents its priority. (These priorities can be used to implement various levels of service in networks with QoS guarantees.) Only one packet can be transmitted in any time slot, so, if the system is overloaded, some packets will inevitably miss their deadlines and be dropped. In this scenario, the natural objective is to compute a transmission schedule that maximizes the total weight of packets which are successfully transmitted. In the literature this problem is also occasionally referred to as *bounded-delay buffer management*, *QoS buffering*, or as a job scheduling problem for unit-length jobs with release times, deadlines, and weights, where the objective is to maximize the weighted throughput.

The problem is inherently online, with the scheduling decisions made without the knowledge of future packet arrivals. The central problem concerning PacketScheduling, that has been a subject of intensive study since 2001, is to determine the optimal competitive ratio of online algorithms, namely the worst-case ratio between the optimum total weight of a schedule (computed by an offline algorithm) and the weight of a schedule computed by a (deterministic) online algorithm.

This paper provides the solution of this open problem by establishing an upper bound of $\phi$ on the competitive ratio for PacketScheduling (where $\phi \approx 1.618$ is the golden ratio), matching the previously known lower bound [16, 3, 22, 10]. Our $\phi$-competitive algorithm PlanM is presented in Section 4. The basic idea underlying our algorithm is relatively simple. It is based on the concept of the *plan*, which, at any given time $t$, is the maximum-weight subset of pending packets that can be feasibly scheduled in the future (if no other packets arrive); we describe it in Section 3. When some packet $p$ from the plan is chosen to be scheduled at time $t$, it will be replaced in the plan by some other packet $\varrho$. The algorithm chooses $p$ to maximize an appropriate linear combination of $w_p$ and $w_\varrho$. For technical reasons, it also makes additional changes in the plan, adjusting deadlines and weights of some packets. While the algorithm itself is not complicated, its competitive analysis given in Section 5, is quite intricate. It relies on showing a bound on amortized gain at each step, using a potential function, which quantifies the advantage of the algorithm over the adversary in future steps, and on maintaining an invariant that allows us to control decreases of the potential function.

**Past work.** The PacketScheduling problem was first introduced independently by Hajek [16] and Kesselman *et al.* [19], who both gave a proof that the greedy algorithm (that always schedules the heaviest packet) is 2-competitive. Hajek's paper also contained a proof of a lower bound of $\phi \approx 1.618$ on the competitive ratio. The

*Charles University, Prague, Czech Republic and University of Warwick, UK. Email: vesely@iuuk.mff.cuni.cz.
†University of California at Riverside, USA. Email: marek@cs.ucr.edu.
‡University of Wrocław, Poland. Email: lje@cs.uni.wroc.pl.
§Charles University, Prague, Czech Republic. Email: sgall@iuuk.mff.cuni.cz.

same lower bound was later discovered independently by Andelman *et al.* [3, 22] and also by Chin *et al.* [10] in a different, but equivalent setting. Improving over the greedy algorithm, Chrobak *et al.* [11, 12] gave an online algorithm with competitive ratio 1.939. This was subsequently improved to 1.854 by Li *et al.* [21], and to 1.828 by Englert and Westermann [14], which, prior to the present paper, has been the best upper bound known.

Algorithms with ratio $\phi$ have been developed for several restricted variants of PacketScheduling. Li *et al.* [20] (see also [18]) gave a $\phi$-competitive algorithm for the case of *agreeable* deadlines, which consists of instances where the deadline ordering is the same as the ordering of release times. Another well-studied case is that of *$s$-bounded instances*, where each packet's deadline is within at most $s$ steps from its release time. A $\phi$-competitive algorithm for 2-bounded instances was given by Kesselman *et al.* [19]. This bound was later extended to 3-bounded instances by Chin *et al.* [9] and to 4-bounded instances by Böhm *et al.* [8]. The work of Bienkowski *et al.* [6] provides an upper bound of $\phi$ (in a somewhat more general setting) for the case where packet weights increase with respect to deadlines. (It should be noted that the lower bound of $\phi$ applies to instances that are 2-bounded, which implies agreeable-deadlines, and have increasing weights.) In *$s$-uniform instances*, the deadline of each packet is *exactly* $s$ steps from its release time, which also implies agreeable deadlines. The lower bound of $\phi$ in [16, 10] does not apply to $s$-uniform instances; as shown by Chrobak *et al.* [12], for 2-uniform instances ratio $\approx 1.377$ is optimal.

Randomized online algorithms for PacketScheduling have been studied as well, although the gap between the upper and lower bounds for the competitive ratio remains quite large. The best upper bound is $\approx 1.582$ [4, 9, 7, 17], and it applies even to the adaptive adversary model. For the adaptive adversary, the best lower bound is $\approx 1.33$ [7], while for the oblivious adversary it is 1.25 [10].

Kesselman *et al.* [19] originally proposed the problem in the setting with integer bandwidth $m \geq 1$, which means that $m$ packets are sent in each step. For any $m$ they proved that the greedy algorithm is 2-competitive and that there is a $\phi$-competitive algorithm for 2-bounded instances [19]. Later, Chin *et al.* [9] gave an algorithm with ratio that tends to $\frac{e}{e-1} \approx 1.582$ for $m \to \infty$. The best lower bound for any $m$, also due to Chin *et al.* [9], equals 1.25 and holds even for randomized algorithms against the oblivious adversary. Observe that any upper bound for bandwidth 1 implies the same upper bound for an arbitrary $m$, by simulating an online algorithm for bandwidth 1 on an instance where each step is subdivided into $m$ smaller steps. Hence, our algorithm in Section 4 is $\phi$-competitive for any $m$, which improves the current state of art for any $m < 13$.

There is a variety of other packet scheduling problems related to PacketScheduling. The semi-online setting with lookahead was proposed in [8]. A relaxed variant of PacketScheduling in which only the ordering of deadlines is known, but not their exact values, was studied in [5], where a lower bound higher than $\phi$ was shown. In the FIFO model (see, for example, [2, 19]), packets do not have deadlines, but the switch has a buffer that can only hold $B$ packets, and packets must be transmitted in the first-in-first-out order. More information about PacketScheduling and related scheduling problems can be found in a survey paper by Goldwasser [15].

## 2 Preliminaries

**The online PacketScheduling problem.** The instance of PacketScheduling is specified by a set of *packets*, with each packet $p$ represented by a triple $(r_p, d_p, w_p)$, where integers $r_p$ and $d_p \geq r_p$ denote the *release time* and *deadline* (or *expiration time*) of $p$, and $w_p \geq 0$ is the *weight* of $p$. (To avoid double indexing, we sometimes use notation $w(p)$ to denote $w_p$ and $d(p)$ for $d_p$.) Time is discrete, with time units represented by consecutive integers that we refer to as *time slots* or *steps*. In a feasible transmission schedule, a subset of packets is transmitted. Only one packet can be transmitted in each time step, and each packet $p$ can only be transmitted in one slot in the interval $[r_p, d_p]$. The objective is to compute a schedule whose total weight of transmitted packets (also called its *profit*) is maximized.

In the online variant of PacketScheduling, which is the focus of our work, the algorithm needs to compute the solution incrementally over time. At any time step $t$, packets with release times equal to $t$ are revealed and added to the set of pending packets (that is, those that are already released, but not yet expired or transmitted). Then the algorithm needs to choose one pending packet to transmit in slot $t$. As this decision is made without the knowledge of packets to be released in future time steps, such an online algorithm cannot, in general, be guaranteed to compute an optimal solution. The quality of the schedules it computes can be then quantified using competitive analysis. We say that an online algorithm $\mathcal{A}$ is *$c$-competitive* if, for each instance, the optimal profit (computed offline) is at most $c$ times the profit of the schedule computed by $\mathcal{A}$.

**Useful assumptions.** We make two assumptions about our problem without loss of generality.

(UA1) We assume that at each step $t$ and for each $\tau \geq t$ (up to a certain large enough limit), there is a pending packet with deadline $\tau$. This can be achieved by releasing, at time $t$, a virtual 0-weight packet with deadline $\tau$, for each $\tau \geq t$.

(UA2) We also assume that all packets have different weights. Any instance can be transformed into an instance with distinct weights through infinitesimal perturbation of the weights, without affecting the competitive ratio. The 0-weight packets from the previous assumption thus, in fact, have an infinitesimal positive weight. The purpose of this assumption is to facilitate consistent tie-breaking, in particular uniqueness of plans (to be defined shortly).

## 3 Plans

Consider an execution of an online algorithm $\mathcal{A}$. At any time $t$, $\mathcal{A}$ will have a set of pending packets. We now discuss properties of these pending packets and introduce the concept of a plan.

The set of packets pending at a time $t$ has a natural ordering, called the *canonical ordering* and denoted $\prec$, which orders packets in non-decreasing order of deadlines, breaking ties in favor of heavier packets. Formally, for two pending packets $x$ and $y$, define $x \prec y$ iff $d_x < d_y$ or $d_x = d_y$ and $w_x > w_y$. The *earliest-deadline packet* in some subset $X$ of pending packets is the packet that is the first in the canonical ordering of $X$. Similarly, the *latest-deadline packet* in $X$ is the last packet in the canonical ordering of $X$.

A subset $X$ of pending packets is called *feasible* if the packets in $X$ can be scheduled in future time slots $t, t+1, \ldots$, meeting their deadlines. Using a standard exchange argument, if $X$ is feasible, then any schedule of $X$ can be converted into its *canonical schedule*, in which the packets from $X$ are assigned to the slots $t, t+1, \ldots$ in the canonical order.

For each slot $\tau \geq t$, let $X_{\leq\tau} = \{j \in X : d_j \leq \tau\}$ be the subset of $X$ consisting of packets with deadline at most $\tau$, and define

$$\mathsf{pslack}(X, \tau) = (\tau - t + 1) - |X_{\leq\tau}|;$$

note that $\tau - t + 1$ is the number of slots in $[t, \tau]$. For convenience, we also allow $\tau = t - 1$ and assume that $\mathsf{pslack}(X, t-1) = 0$. Observe that $X$ is feasible if and only if $\mathsf{pslack}(X, \tau) \geq 0$ for each $\tau \geq t$: If $X$ is feasible then in its schedule determined by the canonical order, for each $\tau \geq t$ all packets in $X_{\leq\tau}$ are scheduled in $[t, \tau]$; thus $|X_{\leq\tau}| \leq \tau - t + 1$. And vice versa, the condition that $\mathsf{pslack}(X, \tau) \geq 0$ for each $\tau \geq t$ implies that in the canonical schedule all packets meet their deadlines.

The collection of feasible subsets of pending packets forms a matroid. This implies that the maximum-weight feasible subset of pending packets, that we call a *plan*, can be found by the following greedy algorithm: Initially, let $P$ be an empty set. For each pending packet $j$ in order of decreasing weights, if $\mathsf{pslack}(P \cup \{j\}, \tau) \geq 0$ for all $\tau \geq t$, then add $j$ to $P$. At the end, $P$ is the plan.

Assumption (UA2) about different weights implies that the plan $P$ computed above is unique. We typically use letters $P, Q, \ldots$ to denote plans. Note that in a plan we do not assign packets to time slots, that is, a plan is not a schedule. A plan has at least one schedule, but in general it may have many. (In the literature, such scheduled plans are sometimes called *optimal provisional schedules*.)

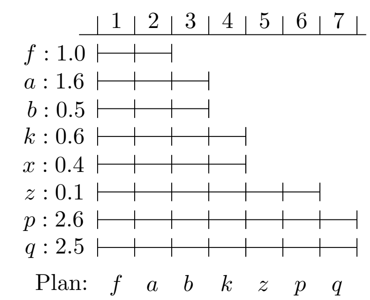


Figure 1: An example of an instance with plan $P = \{f, a, b, k, z, p, q\}$ and its canonical schedule.

We briefly describe the structure of plan $P$ at time $t$. Slot $\tau \geq t$ is called *tight* in $P$ if $\mathsf{pslack}(P, \tau) = 0$. According to our convention, we also consider $t - 1$ to be a tight slot. If the tight slots of $P$ are $t_0 = t - 1 < t_1 < t_2 < \cdots$, then for each $i \geq 1$ the time interval $S_i = (t_{i-1}, t_i] = \{t_{i-1}+1, t_{i-1}+2, \ldots, t_i\}$ is called a *segment* of $P$. In words, the tight slots divide the plan into segments, each starting right after a tight slot and ending at (and including) the next tight slot. The significance of a segment $S_i$ is that in any schedule of $P$ all packets in $P$ with deadlines in $S_i$ must be scheduled in this segment. Thus, slightly abusing terminology, we occasionally think of each $S_i$ as a set of packets, namely the packets in $P$ that must be scheduled in $S_i$. Within a segment, packets can be permuted, although only in some restricted ways. In particular, the first slot of a segment may contain any packet from that segment.

For a plan $P$ and a slot $\tau \geq t$, let $\mathsf{nextts}(P, \tau)$ be the earliest tight slot $\tau' \geq \tau$ (which exists by Assumption (UA1)), and let $\mathsf{prevts}(P, \tau)$ be the latest tight slot $\tau' < \tau$ (recall that $t - 1$ is a tight slot).

The notion that will be crucial in the design of our $\phi$-competitive algorithm is the minimum weight of a packet in the plan that can be scheduled in some slot between the current time and a slot $\tau$. For a plan $P$ at time $t$ and a slot $\tau \geq t$, define

$$\mathsf{minwt}(P, \tau) = \min\{w_\ell : \ell \in P \text{ and } d_\ell \leq \mathsf{nextts}(P, \tau)\}.$$

By definition, all slots $\tau$ in a segment have the same value of $\mathsf{minwt}(P, \tau)$. Also, for a given plan $P$ at time

$t$, if $a \notin P$ then $w_a < \mathsf{minwt}(P, d_a)$, and the function $\mathsf{minwt}(P, \tau)$ is monotonely non-increasing for $\tau \geq t$.

To analyze how the plan changes over time, we divide each step $t$ into a sequence of *events*. First we have events representing packet arrivals, with each packet released at time $t$ being added to the set of pending packets. The last event represents scheduling a packet for transmission and incrementing the current time to $t + 1$. The matroid property implies that at most one other packet in the plan changes after each event (not counting the scheduled packet in a scheduling event). We outline these changes below; formal proofs will appear in the full version of this paper.

**Packet arrival.** Let $t$ be the current time, $P$ be the current plan, and suppose that $j$ is a new packet arriving at time $t$. As $j$ is added to the set of pending packets, the plan needs to be updated accordingly. Define $f \in P$ to be the packet with $w_f = \mathsf{minwt}(P, d_j)$, that is the lightest packet in $P$ with $d_f \leq \mathsf{nextts}(P, d_j)$. If $w_j < w_f$, then $j$ is not added to the plan and the plan stays the same, while if $w_j > w_f$, then $j$ is added to the plan and $f$ is forced out, i.e., the new plan is $Q = P \cup \{j\} \setminus \{f\}$. In the latter case, it is interesting to see how the values of $\mathsf{pslack}()$ and the segments change:

- If $d_j \geq d_f$, then $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) + 1$ for $\tau \in [d_f, d_j)$. Therefore, all tight slots in $[d_f, d_j)$ are no longer tight and the segments containing $d_f$ and $d_j$ and all segments in-between get merged into one segment of $Q$.
- If $d_j < d_f$, then $d_f$ and $d_j$ are in the same segment of $P$ and $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ for $\tau \in [d_j, d_f)$. Thus there may be new tight slots in $[d_j, d_f)$, resulting in new segments.

In both cases, the values of $\mathsf{pslack}()$ remain the same for other slots. Moreover, $\mathsf{minwt}(Q, \tau) \geq \mathsf{minwt}(P, \tau)$ holds for any slot $\tau \geq t$.

**Scheduling a packet.** Next, suppose that $P$ is the plan at time $t$ after all packets arriving at time $t$ are aleady added to the set of pending packets. Suppose that we decide to schedule a packet $p \in P$ at time $t$. Let $Q$ be the new plan after $p$ is scheduled and the current time is incremented to $t + 1$.

If $p$ is from segment $S_1$ of $P$, then $Q = P \setminus \{p\}$. In this case $\mathsf{pslack}(\tau)$ decreases by 1 for $\tau \in [t+1, d_p)$ and remains unchanged for $t \geq d_p$. This implies that new tight slots may appear before $d_p$, i.e., the first segment may get divided into more segments. Also, $\mathsf{minwt}(\tau)$ does not decrease for any $\tau \geq t + 1$.

The more interesting case is when $p$ is from a later segment than $S_1$. Let $\omega$ be the lightest packet in $S_1$ and let $\varrho$ be the heaviest pending packet not in $P$ that satisfies $d_\varrho > \mathsf{prevts}(P, d_p)$. Using the matroid property of the feasible sets of packets at time $t + 1$ and the structure of the plan it is possible to prove that $Q = P \setminus \{p, \omega\} \cup \{\varrho\}$. In this case:

- $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ for $\tau \in [t + 1, d_\omega)$. There may be new tight slots in the interval $[t + 1, d_\omega)$, resulting in new segments.
- If $d_\varrho \geq d_p$, then $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) + 1$ for $\tau \in [d_p, d_\varrho)$. Here, all segments that overlap $[d_p, d_\varrho)$ are merged into one segment of $Q$.
- If $d_\varrho < d_p$, then $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ for $\tau \in [d_\varrho, d_p)$. Thus new tight slots may appear in $[d_\varrho, d_p)$, resulting in new segments.

For slots $\tau \geq t + 1$ not covered by the cases above, the value of $\mathsf{pslack}(\tau)$ does not change. Unlike for packet arrivals, after a packet scheduling event some values of $\mathsf{minwt}(\tau)$ may decrease, either due to $\varrho$ being included in $Q$ or as a side-effect of segments being merged.

Let $P$ be the plan at time $t$. For each $j \in P$ we define the *substitute packet of* $j$, denoted $\mathsf{sub}(P, j)$, as follows. If $j \in S_1$, then $\mathsf{sub}(P, j) = \omega$, where $\omega$ is the lightest packet in $S_1$. If $j \notin S_1$, then $\mathsf{sub}(P, j)$ is the heaviest pending packet $\varrho \notin P$ that satisfies $d_\varrho > \mathsf{prevts}(P, d_j)$ (it exists by assumption (UA1)).

By definition, all packets in a segment of $P$ have the same substitute packet. Also, for any $j \in P$ it holds that $w_j \geq w(\mathsf{sub}(P, j))$. This is because for $j \in S_1$ we have $\mathsf{sub}(P, j) = \omega$ and $w_j \geq w_\omega$, while for $j \in P \setminus S_1$ we have $d(\mathsf{sub}(P, j)) > \mathsf{prevts}(P, d_j)$; thus in this case, the set $P - \{j\} \cup \{\mathsf{sub}(P, j)\}$ is feasible and the optimality of $P$ implies that $w_j \geq w(\mathsf{sub}(P, j))$.

## 4 Online Algorithm

**Intuitions.** For profit maximization problems, the challenge in the online setting is to balance the immediate profit against future profits. Let $P$ be the plan at step $t$. Consider the greedy algorithm for $\mathsf{PacketScheduling}$, which at time $t$ schedules the heaviest pending packet $h$ (which is necessarily in $P$). As a result, in the next step $h$ would be replaced in the plan by its substitute packet $\varrho_h = \mathsf{sub}(P, h)$, which could be very light, possibly $w(\varrho_h) \approx 0$. Suppose that there is another packet $g$ in the plan with $w_g \approx w_h$ whose substitute packet $\varrho_g = \mathsf{sub}(P, g)$ is quite heavy, say $w(\varrho_g) \approx w_g$. Thus instead of $h$ we can schedule $g$ at time $t$, gaining about as much as from $h$ in step $t$, but with essentially no decrease in future profit. This example indicates that a reasonable strategy would be to choose a packet $p$ based both on its weight and the weight of its substitute packet. Following this intuition, our algorithm chooses $p$ that maximizes $w_p + \phi \cdot w(\mathsf{sub}(P, p))$.

As it turns out, the above strategy for choosing $p$ does not, by itself, guarantee $\phi$-competitiveness. The analysis of special cases and an example where this simple approach fails leads to the second idea behind our algorithm. The difficulty is related to how the values of $\mathsf{minwt}(\tau)$, for a fixed $\tau$, vary while the current time $t$ increases. We were able to show $\phi$-competitiveness of the above strategy for certain instances where $\mathsf{minwt}(\tau)$ monotonely increases as $t$ grows from 0 to $\tau$. We call this property *slot-monotonicity.* To extend it to instances where slot monotonicity does not hold, the idea is then to simply *force* it to hold by decreasing deadlines and increasing weights of some packets in the new plan. (These weight increases will be accounted for appropriately in the analysis.)

**Notation.** To avoid ambiguity, we will index various quantities used by the algorithm with the superscript $t$ that represents the current time. This includes weights and deadlines of some packets, since, as described above, these might change over time.

- We use notation $w^t_p$ and $d^t_p$ for the weight and the deadline of packet $p$ in step $t$, before a packet is scheduled. (Our algorithm only changes weights and deadlines when scheduling a packet, so they are not affected by packet arrivals.) To avoid double subscripts, we occasionally write $w^t(p)$ and $d^t(p)$ instead of $w^t_p$ and $d^t_p$. By $w^0_p$ we denote the original weight of packet $p$. We may omit $t$ in these notations when $t$ is implied from context.
- $P^t$ is the plan at time $t$ after all packets $j$ with $r_j = t$ arrive and before a packet is scheduled. By $S_1$ we denote the first segment of $P^t$.
- $\omega$ is the lightest packet in segment $S_1$ of $P^t$.
- We use $\mathsf{sub}^t(p)$ to denote $\mathsf{sub}(P^t, p)$ and similarly for $\mathsf{minwt}^t(\tau), \mathsf{nextts}^t(\tau)$, and $\mathsf{prevts}^t(\tau)$.

**Algorithm 1** Algorithm $\mathsf{PlanM}(t)$

1: schedule $p \in P^t$ maximizing $w^t_p + \phi \cdot w^t(\mathsf{sub}^t(p))$
2: **if** $p \notin S_1$ (first segment of $P^t$) **then** ▷ "leap step"
3: $\quad \varrho \leftarrow \mathsf{sub}^t(p)$
4: $\quad w^{t+1}_\varrho \leftarrow \mathsf{minwt}^t(d^t_\varrho)$ ▷ increase $w_\varrho$
5: $\quad \gamma \leftarrow \mathsf{nextts}^t(d^t_\varrho)$ and $\tau_0 \leftarrow \mathsf{nextts}^t(d^t_p)$
6: $\quad i \leftarrow 0$ and $h_0 \leftarrow p$
7: $\quad$ **while** $\tau_i < \gamma$ **do**
8: $\quad\quad i \leftarrow i + 1$
9: $\quad\quad h_i \leftarrow$ heaviest packet in $P^t$ s.t. $d^t_{h_i} \in (\tau_{i-1}, \gamma]$
10: $\quad\quad \tau_i \leftarrow \mathsf{nextts}^t(d^t_{h_i})$
11: $\quad\quad d^{t+1}_{h_i} \leftarrow \tau_{i-1}$ and $w^{t+1}_{h_i} \leftarrow \max(w^t_{h_i}, \mathsf{minwt}^t(\tau_{i-1}))$
12: $\quad k \leftarrow i$ ▷ final value of $i$

For a pending packet $j$, if $w^{t+1}_j$, resp. $d^{t+1}_j$ is not explicitly set in the algorithm, then $w^{t+1}_j \leftarrow w^t_j$, resp. $d^{t+1}_j \leftarrow d^t_j$, i.e., the weight, resp. the deadline remains the same by default.

Let $p$ be the packet sent by $\mathsf{PlanM}$ in step $t$. If $p$ is in the first segment $S_1$ of $P^t$, the step is called a *greedy step.* Otherwise (if $p \notin S_1$), the step is called a *leap step*, and then $\varrho = \mathsf{sub}^t(p)$ is the heaviest pending packet $\varrho \notin P^t$ with $d^t_\varrho > \mathsf{prevts}^t(d^t_p)$. We will further consider two types of leap steps. If $p$ and $\varrho$ are in the same segment (formally, when $\tau_0 = \gamma$, or equivalently, $k = 0$), then this leap step is called a *simple leap step.* If $\varrho$ is in a later segment than $p$ (that is, when $\gamma > \tau_0$, which is equivalent to $k > 0$) then this leap step is called an *iterated leap step.*

As all packets in the segment of $P^t$ containing $p$ have the same substitute packet $\mathsf{sub}^t(p)$, $p$ must be the heaviest packet in its segment. Furthermore, $p$ is not too light compared to the heaviest pending packet $h$; specifically, we have that $w_p \geq w_h/\phi^2$, which can be derived from the choice of $p$ in line 1.

**Slot-monotonicity.** Our goal is to maintain the slot-monotonicity property, i.e., to ensure that for any fixed slot $\tau$ the value of $\mathsf{minwt}^t(\tau)$ does not decrease as the current time $t$ progresses from 0 to $\tau$. For this reason, we need to increase the weight of the substitute packet $\varrho$ in each leap step (as $w^t_\varrho < \mathsf{minwt}^t(d^t_\varrho)$), which is done in line 4. For the same reason, we also need to adjust the deadlines and weights of the packets $h_i$, which is done in line 11. The deadlines of $h_i$'s are decreased to make sure that the segments between $\delta = \mathsf{prevts}^t(d^t_p)$ and $\gamma$ do not merge (as merging could cause a decrease of some values of $\mathsf{minwt}^t(\tau)$). These deadline changes can be thought of as a sequence of substitutions, where $h_1$ replaces $p$ in the segment of $P$ ending at $\tau_0$, $h_2$ replaces $h_1$, etc., and finally, $\varrho$ replaces $h_k$ in the segment ending at $\gamma$. We sometimes refer to this process as a "shift" of the $h_i$'s. Then, if the weight of some $h_i$ is too low for its new segment, it is increased to match the earlier minimum of that segment, that is $\mathsf{minwt}^t(\tau_{i-1})$.

We briefly outline changes in the plan after a leap step; the details and formal proofs are omitted. By the definition in line 9 and the while loop condition in line 7, we have that $w^t_p = w^t_{h_0} > w^t_{h_1} > w^t_{h_2} > \cdots > w^t_{h_k} > w^t_\varrho$ and that $h_k$'s deadline is in the segment of $P$ ending at $\gamma$, that is $\mathsf{prevts}^t(d^t_\varrho) < d^t_{h_k} \leq \gamma$.

Let $P = P^t$ and let $Q$ be the plan after $p$ is scheduled, the time is incremented to $t+1$, and weights and deadlines are changed as in the algorithm. Let $\overline{Q}$ be the plan after $p$ is scheduled and the time is incremented, but before the algorithm adjusts weights and deadlines. As discussed in Section 3, after $p$ from a later segment is scheduled, the plan is $\overline{Q} = P \setminus \{p, \omega\} \cup \{\varrho\}$, where $\varrho = \mathsf{sub}^t(p)$. Observe that increasing the weight of a packet in the plan does not change the plan. Moreover, an analysis of the changes of $\mathsf{pslack}()$ values yields that decreasing the deadlines of $h_1, h_2, ..., h_k$ (in line 11) does not change the plan, so $Q = P \setminus \{p, \omega\} \cup \{\varrho\}$ holds even in an iterated leap step, that is $Q = \overline{Q}$.

The decrease of the deadlines ensures that any tight slot of $P$ is tight in $Q$ as well. This property, together with the increase of the weights, allows us to prove that $\mathsf{minwt}^t(\tau)$ does not decrease in a leap step.

Lemma 4.1. *Let $P$ be the current plan in step $t$ just before an event of either arrival of a new packet, or scheduling a packet (and incrementing the current time), and let $Q$ be the plan after the event. Then $\mathsf{minwt}(Q,\tau) \geq \mathsf{minwt}(P,\tau)$ for any $\tau > t$ and also for $\tau = t$ in the case of packet arrival.*

*Hence, in the computation of Algorithm* PlanM*, for any fixed $\tau$, function $\mathsf{minwt}^t(\tau)$ is non-decreasing in $t$ as $t$ grows from $0$ to $\tau$.*

**Comparison to previous algorithms.** Our algorithm shares some broad features with known algorithms in the literature. Some prior algorithms used the notion of *optimal provisional schedules*, which coincides with our concept of canonically ordered plans. For example, the $\phi$-competitive algorithm MG for instances with agreeable deadlines by Li *et al.* [18] (see also [20]) transmits packets from the plan only, either the heaviest packet $h$ or the earliest-deadline packet $e$. The same authors [21] later designed a modified algorithm called DP (using memory) that achieves competitive ratio $3/\phi \approx 1.854$ for arbitrary instances.

Our approach is similar to that of Englert and Westermann [14], who designed a 1.893-competitive memoryless algorithm and an improved 1.828-competitive variant with memory. Both their algorithms are based on the notion of *suppressed packet* $\mathsf{supp}(P,p)$, for a packet $p$ in the plan $P$, which, in our terminology, is the same as the substitute packet $\mathsf{sub}(P,p)$ if $p$ is not in the first segment. However, the two concepts differ for packets $p$ in the first segment. The memoryless algorithm in [14] identifies a packet $m$ of maximum "benefit", which is measured by an appropriate linear combination of $w_m$ and $w(\mathsf{supp}(P,m))$, and sends either $m$ or $e$ (the earliest-deadline packet in the plan), based on the relation between $w_e$ and the benefit of $m$. The algorithm with memory in [14] extends this approach by comparing $m$'s benefit to $e$'s "boosted weight" $\max(w_e, \delta(t))$, where $t$ is the current step and $\delta(\tau)$ is the maximum value of $\mathsf{minwt}(P^{t'},\tau)$ over $t' < t$.

Our algorithm involves several new ingredients that are critical to establishing competitive ratio $\phi$. First, our analysis relies on full characterization of the evolution of the plan over time, in response to packet arrivals and scheduling events (briefly described in Section 3). Two, we introduce a new objective function $w^t_p + \phi \cdot w^t(\mathsf{sub}^t(p))$ for selecting a packet $p$ for scheduling. This function is based on a definition of substitute packets, $\mathsf{sub}(P,p)$, that accurately reflects the changes in the plan following scheduling events, including the case when $p$ is in the first segment. Three, we introduce the concept of slot monotonicity, and devise a way for the algorithm to maintain it over time. This property is very helpful in keeping track of the optimal profit. Last but not least, we introduce a potential function, that captures the "advantage" of the algorithm over the adversary regarding future time steps.

## 5 Competitive Analysis

Let $\mathsf{ALG}$ be the schedule of $\mathsf{PlanM}$ for an instance of $\mathsf{PacketScheduling}$ under consideration, and let $\mathsf{OPT}$ be a fixed optimal schedule for this instance (actually, $\mathsf{OPT}$ can be any schedule for this instance). Our overall goal is to show that $\phi \cdot w^0(\mathsf{ALG}) \geq w^0(\mathsf{OPT})$. (Recall that $w^0_j$ denotes the original weight of packet $j$).

**5.1 Adversary Schedule and Shadow Packets** In the analysis we will actually work with the *adversary schedule* $\mathsf{ADV}$ that serves as a mechanism for keeping track of future adversary's gain associated with the already-released packets from $\mathsf{OPT}$. (Abusing notation, we use $\mathsf{ADV}$ to also denote the set of packets in the adversary schedule.) Roughly, at each step $t$, $\mathsf{ADV}$ is meant to consist of the already-released packets from $\mathsf{OPT}$ that have not yet been scheduled. So initially $\mathsf{ADV}$ is empty, and later whenever a packet $j$ arrives and $j \in \mathsf{OPT}$ then we add $j$ to $\mathsf{ADV}$ to the slot in which $j$ is in $\mathsf{OPT}$. At each step $t$, we will also remove packet $\mathsf{ADV}[t]$ from $\mathsf{ADV}$ (and the adversary gains its weight).

However, in addition, during the course of the analysis we will also occasionally make modifications to $\mathsf{ADV}$ by replacing some packets in $\mathsf{ADV}$ by lighter or equal-weight packets, either real packets (including those from Assumption (UA1)) or fictitious *shadow packets*, described below. As a result of such changes, at any time $t$, even if $\mathsf{OPT}[\tau]$ contains a packet released at or before time $t$, the packet in $\mathsf{ADV}[\tau]$ may be different.

Shadow packets are in essence just an accounting trick: they represent deposits of profit, to be collected when the current time reaches their associated time slot. When a shadow packet $s$ is created and added to slot $\tau_s$ in $\mathsf{ADV}$ it satisfies $w_s \leq \mathsf{minwt}^t(\tau_s)$. From now on it is tied to its slot and never changes. Therefore, by Lemma 4.1, its weight does not exceed $\mathsf{minwt}^t(\tau_s)$, until it is eventually scheduled by the adversary when the current time $t$ reaches $\tau_s$. Further, shadow packets exist only in $\mathsf{ADV}$ — they are not pending for the algorithm at any time. We thus do not need to impose a canonical order on them or specify their release times and deadlines. They are also exempt from assumption (UA2). Shadow packets are introduced in the course of the analysis to ensure that certain invariants (defined below) are preserved when a new packet arrives or when a packet is scheduled.

Replacement by real packets in $\mathsf{ADV}$ may occur in an iterated leap step when, under some circumstances, we replace a packet $h_i \in \mathsf{ADV}$ by $h_{i+1}$, which is always lighter than $h_i$. These replacements need to be done carefully to avoid packet duplication. (As a forward reference, we note that this replacement happens only in Case M.ii in Section 5.5.5.)

Note that real packets in $\mathsf{ADV}$ have their current weights $w^t$ and deadlines $d^t$, i.e., the same as for the algorithm and not necessarily equal to the original values. (We remark that there will be no real packets in $\mathsf{ADV}$ that are not pending for the algorithm.) This implies that when the algorithm increases the weight of a packet $g$ which is present in $\mathsf{ADV}$, the total weight of $\mathsf{ADV}$ increases by the same amount. In fact, if this happens, we replace $g$ by another packet in $\mathsf{ADV}$, as described above, and we do not place $g$ into another slot of $\mathsf{ADV}$ in the current step. (However, $g$ may be readded to $\mathsf{ADV}$ in a later step with its new weight.)

Regarding decreasing the deadlines of $h_i$'s in an iterated leap step, to guarantee that no packet is in $\mathsf{ADV}$ in a slot after its current deadline, we also replace each $h_i$ either by a shadow packet or by $h_{i+1}$. In the latter case, as the new deadline of $h_{i+1}$ is $\tau_i \geq d^t_{h_i}$ and as $h_{i+1}$ is added to the former slot of $h_i$ (which is not after $d^t_{h_i}$), we guarantee that $h_{i+1}$ is not after its new deadline in $\mathsf{ADV}$.

The following invariant, maintained throughout the analysis, captures properties of the packets in $\mathsf{ADV}$ that will be crucial for our argument (see Figure 2):

**(InvA)** At each step $t$, $\mathsf{ADV}$ consists of two types of packets:

- Packets in $\mathsf{ADV} \cap P^t$. Each such packet $g$ is in $\mathsf{ADV}$ in a slot in $[t, d^t_g]$.
- Packets in $\mathsf{ADV} \setminus P^t$. All these packets are shadow packets, with properties described above; in particular each shadow packet $s = \mathsf{ADV}[\tau]$ is not pending for the algorithm and satisfies $w_s \leq \mathsf{minwt}^t(\tau)$.

After each event we change the adversary schedule $\mathsf{ADV}$ so that invariant (InvA) is preserved. Sometimes, it will be convenient to do the analysis in stages, in each stage considering an interval of time slots. We say that invariant (InvA) holds for an interval $S$ of slots if (InvA) holds for all packets in $\mathsf{ADV}$ with deadlines in $S$.

**Amortized analysis.** We bound the competitive ratio via amortized analysis, using a combination of three accounting techniques:

- In leap steps, when the algorithm increases weights of some packets (the substitute packet and some $h_i$'s), we charge it a "penalty" equal to $\phi$ times the total weight increase.
- We use a potential function (see Section 5.3), which quantifies the advantage of the algorithm over the adversary in future steps.
- As mentioned earlier, in some situations we replace packets in $\mathsf{ADV}$ by lighter packets. If this happens, we add the appropriate "credit" (equal to the weight decrease) to the adversary's gain.

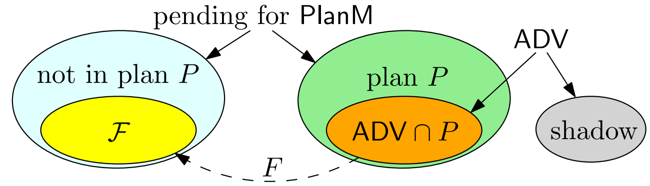


Figure 2: The sets of packets in the competitive analysis. Set $\mathcal{F}$ and bijection $F : \mathsf{ADV} \cap P \to \mathcal{F}$ are introduced in Section 5.2.

To ensure that the current plan $P^t$ and the adversary schedule $\mathsf{ADV}$ satisfy desired structural properties, we maintain two invariants: (InvA), defined above, and (InvP), that will be introduced in Section 5.2 below.

**5.2 Set $\mathcal{F}$ and Invariant (InvP)** In our analysis we maintain a set $\mathcal{F}$, which is a subset of "forced-out" pending packets, i.e., packets that were ousted from the plan, either as a result of arrivals of other packets or in a leap step. A useful property of $\mathcal{F}$ is that each packet in $\mathcal{F}$ might be useful as a substitute packet.

In our analysis we will maintain the invariant that $|\mathcal{F}| = |\mathsf{ADV} \cap P|$ (where $P$ is the current plan). We also use the following natural bijection $F$ between $\mathsf{ADV} \cap P$ and $\mathcal{F}$: Let $f_1, \ldots, f_\ell$ be all packets in $\mathcal{F}$ in the canonical ordering, i.e., $d_{f_1} \leq d_{f_2} \leq \cdots \leq d_{f_\ell}$ (breaking ties in favor of heavier packets), and let $g_1, \ldots, g_\ell$ be all packets in $\mathsf{ADV} \cap P$, again in the canonical ordering. Then $F(g_i) = f_i$ for all $i$.

For each slot $\tau \geq t$ of the current plan, we define a quantity that will be crucial in our analysis; its name is explained later in this section:

$$\#\mathsf{pairs}(\tau) = |\mathcal{F}_{\leq\tau}| - |(\mathsf{ADV} \cap P)_{\leq\tau}| \,. \tag{5.1}$$

(Recall that if $X$ is a set of pending packets then $X_{\leq\tau} = \{x \in X : d^t_x \leq \tau\}$.)

Throughout the analysis, we will maintain the following important invariant which relates the values of $\mathsf{pslack}()$ and of $\#\mathsf{pairs}()$:

**(InvP)** If $P$ is a plan at time $t$, then for any slot $\tau \geq t$ it holds that $\mathsf{pslack}(P, \tau) \geq \#\mathsf{pairs}(\tau)$.

By expanding the definitions of $\mathsf{pslack}(P, \tau)$ and of $\#\mathsf{pairs}(\tau)$ and rearranging, we get that invariant (InvP) for a slot $\tau$ can equivalently be defined as $|\mathcal{F}_{\le\tau}| + |(P \setminus \mathsf{ADV})_{\le\tau}| \le \tau - t + 1$. Thus, intuitively, this invariant guarantees that if we modify $P$ by replacing any subset of packets $g \in \mathsf{ADV} \cap P$ by the corresponding packets $F(g)$, we obtain a feasible set of pending packets.

Similarly as for invariant (InvA), we make changes in the adversary schedule $\mathsf{ADV}$ and set $\mathcal{F}$ to preserve invariant (InvP). In some cases, we modify these sets in stages, each stage involving modifications that affect an interval of time slots. We will say that invariant (InvP) holds for an interval $S$ of time slots (e.g., a segment of the current plan) if (InvP) holds for any $\tau \in S$.

**More about pairs.** Next, we give an intuitive view of bijection $F$ and invariant (InvP) and then we state some corollaries of this invariant. Note that bijection $F =: \mathsf{ADV} \cap P \to \mathcal{F}$ can equivalently be viewed as a set of pairs $(f_i, g_i)$, $i = 1, \dots, \ell$, such that $f_i = F(g_i)$; we will work with both these pairs and $F$.

We classify the pairs and define their *d-intervals* as follows: A pair $(f, g)$ is *positive* if $d_f < d_g$, *negative* if $d_f > d_g$, and otherwise, if $d_f = d_g$, the pair is *neutral*. The $d$-interval of a pair $(f, g)$ is $[d_f, d_g)$ if the pair is positive, and $[d_g, d_f)$ otherwise. Note that the $d$-interval of a pair is always left-closed and right-open. Moreover, a pair contains a slot $\tau$ if its $d$-interval contains $\tau$, i.e., if $d_f \le \tau < d_g$ for a positive pair, and if $d_g \le \tau < d_f$ for a negative pair. A neutral pair does not contain any slot as the corresponding $d$-interval is empty.

By the definition of $F$, the pairs are *agreeable*, i.e., for any two pairs $(f, g)$ and $(f', g')$, if $d_f < d_{f'}$, then $d_g \le d_{g'}$. Indeed, if $d_f < d_{f'}$, then $f$ is before $f'$ in the canonical ordering of $\mathcal{F}$, thus also $g$ is before $g'$ in the canonical ordering of $\mathsf{ADV} \cap P$ and $d_g \le d_{g'}$ follows. Similarly, a positive pair does not *overlap* with a negative pair $(f', g')$, i.e., there is no slot contained in both pairs.

Recall that $\#\mathsf{pairs}(\tau) = |\mathcal{F}_{\le\tau}| - |(\mathsf{ADV} \cap P)_{\le\tau}|$. Observe that $\#\mathsf{pairs}(\tau)$ equals the number of positive pairs containing $\tau$ minus the number of negative pairs containing $\tau$. As positive and negative pairs do not overlap, $\#\mathsf{pairs}(\tau)$ is either the number of positive pairs containing $\tau$, or minus the number of negative pairs containing $\tau$.

Since $\mathsf{pslack}(\tau)$ is non-negative, an equivalent formulation of invariant (InvP) is that $\mathsf{pslack}(\tau)$ is at least the number of positive pairs containing slot $\tau$. From the invariant it follows that there is no positive pair containing a tight slot, although a negative pair may contain a tight slot. It follows that the $d$-interval of a positive pair is fully contained in a single segment, while the $d$-interval of a negative pair may span several segments.

The important, though simple consequences of invariant (InvP) are summarized in the following lemmas. Namely, we show that each $g$ in $\mathsf{ADV} \cap P$ has a good substitute packet. (The proof of the second lemma is omitted.)

LEMMA 5.1. *Suppose that $f \in \mathcal{F}$, $g \in \mathsf{ADV} \cap P$ and let $f = F(g)$. Then:*
*(a) $d_f > \mathsf{prevts}(P, d_g)$,*
*(b) $w(\mathsf{sub}(P, g)) \ge w_f$, and*
*(c) $w_f < \mathsf{minwt}(P, d_g) \le w_g$.*

*Proof.* (a) Let $\delta = \mathsf{prevts}(P, d_g)$. As $\delta$ is a tight slot, $\mathsf{pslack}(P, \delta) = 0$. Applying (InvP) for $\tau = \delta$ we obtain that $|\mathcal{F}_{\le\delta}| \le |(\mathsf{ADV} \cap P)_{\le\delta}|$, and then the definition of mapping $F()$ implies that $\mathcal{F}_{\le\delta} \subseteq F((\mathsf{ADV} \cap P)_{\le\delta})$. As $g \notin (\mathsf{ADV} \cap P)_{\le\delta}$ and $f = F(g)$, we have $f \notin \mathcal{F}_{\le\delta}$; or, in other words, $d_f > \delta$.

(b) Note that $f \in \mathcal{F}$ is pending, but not in $P$. If $g \in S_1$, then $\mathsf{sub}(P, g) = \omega$ and $w_\omega \ge w_f$ as $\omega$ is heavier than any pending packet not in $P$. Otherwise, by (a) $d_f > \mathsf{prevts}(P, d_g)$ and thus $f$ is a candidate for the substitute packet $\mathsf{sub}(P, g)$, which implies the inequality.

(c) As $f$ is pending, but not in $P$ and as $d_f > \mathsf{prevts}(P, d_g)$ by (a), we have $w_f < \mathsf{minwt}(P, d_g)$. The inequality $\mathsf{minwt}(P, d_g) \le w_g$ follows from $g \in P$.

LEMMA 5.2. *In each step $t$, $|\mathcal{F}_{\le t}| \le 1$, i.e., there is at most one packet $f \in \mathcal{F}$ with $d_f = t$.*

In some cases of the analysis we have situations when a packet $g \in \mathsf{ADV} \cap P$ needs to be removed from $\mathsf{ADV}$ or $P$, forcing us to also remove $f = F(g)$ from $\mathcal{F}$. The next observation shows that this modification preserves invariant (InvP).

LEMMA 5.3. *Suppose that $f \in \mathcal{F}$, $g \in \mathsf{ADV} \cap P$ and let $f = F(g)$. If we remove $f$ from $\mathcal{F}$ and $g$ from $\mathsf{ADV} \cap P$, then:*
*(a) The values of $\#\mathsf{pairs}(\tau)$ change as follows: If $d_f \le d_g$, then $\#\mathsf{pairs}(\tau)$ decreases by 1 for $\tau \in [d_f, d_g)$. On the other hand, if $d_f > d_g$ then $\#\mathsf{pairs}(\tau) \le 0$ for $\tau \in [d_g, d_f)$ both before and after these removals. In both cases, for other slots the value of $\#\mathsf{pairs}(\tau)$ stays the same.*
*(b) Invariant (InvP) remains to hold.*

*Proof.* Claim (b) follows from (a), so it is sufficient to prove (a). First, suppose $d_f \le d_g$. Note that $\#\mathsf{pairs}(\tau)$ remains the same for $\tau \ge d_g$ and for $\tau < d_f$ as both $f$ and $g$ are taken into account before their removals, or none of them, respectively. For $\tau \in [d_f, d_g)$, only $f$ appears in (5.1), thus $\#\mathsf{pairs}(\tau)$ decreases by 1 after we remove $f$.

Next, consider the case $d_f > d_g$. Similarly, $\#\mathsf{pairs}(\tau)$ remains the same for $\tau \geq d_f$ and for $\tau < d_g$. For a slot $\tau \in [d_g, d_f)$, $\#\mathsf{pairs}(\tau)$ increases by 1 as only $g$ was taken into account and not $f$. Since the position of $f$ in the canonical ordering of $\mathcal{F}$ is the same as the position of $g$ in the canonical ordering of $\mathsf{ADV} \cap P$, we get that $|\mathcal{F}_{\leq\tau}| < |(\mathsf{ADV} \cap P)_{\leq\tau}|$, meaning that $\#\mathsf{pairs}(\tau) < 0$ before the removals. It follows that $\#\mathsf{pairs}(\tau) \leq 0$ after the removals.

### 5.3 Potential Function and Overview of the Analysis

Sets $\mathcal{F}$, $\mathsf{ADV}$ and $P$ undergo changes in the course of our analysis, not only when a packet is scheduled, but also when new packets arrive. We thus index these sets not by the current time, but by *events*, introduced earlier in Section 3. Recall that an event is either the arrival of a new packet, or scheduling a packet in step $t$ (together with incrementing the current time). Events are numbered by integers, starting from 0. Let $P_\sigma$ be the plan just before event $\sigma$. Similarly, notations $\mathcal{F}_\sigma$ and $\mathsf{ADV}_\sigma$ represent set $\mathcal{F}$ and the adversary schedule $\mathsf{ADV}$, respectively, right before event $\sigma$. Note that if $\sigma$ is the scheduling event in step $t$, then $P^t = P_\sigma$.

The potential just before event $\sigma$ at time $t$ is:

$$\Psi_\sigma := \tfrac{1}{\phi}\left[\, w^t(P_\sigma) + w^t(\mathcal{F}_\sigma) - w^t(\mathsf{ADV}_\sigma \cap P_\sigma)\,\right]. \tag{5.2}$$

We remark that the potential can equivalently be defined as $\frac{1}{\phi}[w^t(P_\sigma \setminus \mathsf{ADV}_\sigma) + w^t(\mathcal{F}_\sigma)]$, but the above form is more convenient to work with.

**Initial and final state.** At the beginning, we assume that the plan is filled with virtual 0-weight packets, each in a slot equal to its deadline, and none of them scheduled by the adversary. Both set $\mathcal{F}$ and the adversary schedule $\mathsf{ADV}$ are empty, thus invariant (InvP) clearly holds, and $\Psi_0 = 0$. At the end, after all (non-virtual) packets expire, the potential equals 0 as well.

**Adversary gain.** In each step $t$, *the adversary gain*, denoted $\mathsf{advgain}^t$, is defined as the weight of packet $\mathsf{ADV}[t]$ that the adversary schedules in step $t$ *plus* the credit (the difference between old and new weights) for replacing some packets in $\mathsf{ADV}$ by lighter packets. Each packet $j = \mathsf{OPT}[\tau]$ is added to $\mathsf{ADV}[\tau]$ upon its arrival with its original weight, and the adversary gets credit whenever the weight of the packet in $\mathsf{ADV}[\tau]$ is decreased, and also when packet $\mathsf{ADV}[\tau]$ is scheduled when the current time $t$ reaches $\tau$. This implies that $w^0(\mathsf{OPT}) = \sum_t \mathsf{advgain}^t$.

**Amortized analysis.** At the core of our analysis are bounds relating amortized gains of the algorithm and the adversary at each event $\sigma$. If $\sigma$ is the index of a packet arrival event, then we will show the following *packet-arrival inequality*:

$$\Psi_{\sigma+1} - \Psi_\sigma \geq 0\,. \tag{5.3}$$

If $\sigma$ is the index of the scheduling event in a step $t$, then we will show the following *packet-scheduling inequality*:

$$\begin{aligned} &\phi[\, w^t(\mathsf{ALG}[t]) - \Delta^t\text{Weights}\,] + (\Psi_{\sigma+1} - \Psi_\sigma) \\ &\qquad \geq \mathsf{advgain}^t, \end{aligned} \tag{5.4}$$

where $\mathsf{ALG}[t]$ is the packet in slot $t$ in the algorithm's schedule $\mathsf{ALG}$, and $\Delta^t$Weights is the total amount by which the algorithm increases the weights of its pending packets in step $t$.

We prove the packet-arrival inequality in Section 5.4 and the packet-scheduling inequality in Section 5.5. Assuming that these two inequalities hold, we now show our main result.

THEOREM 5.1. *Algorithm PlanM is $\phi$-competitive.*

*Proof.* We show that $\phi\, w^0(\mathsf{ALG}) \geq w^0(\mathsf{OPT})$, which implies the theorem. First, note that the sum of terms $\Psi_{\sigma+1} - \Psi_\sigma$ over all events $\sigma$ equals $\Psi_{T+1} - \Psi_0$, where $\Psi_0 = 0$ is the initial potential and $\Psi_{T+1} = 0$ is the final potential after the last (scheduling) event $T$. So $\sum_\sigma(\Psi_{\sigma+1} - \Psi_\sigma) = 0$. Second, as we noted above, we have $w^0(\mathsf{OPT}) = \sum_t \mathsf{advgain}^t$. Finally, observe that

$$\sum_t [\, w^t(\mathsf{ALG}[t]) - \Delta^t\text{Weights}\,] \leq w^0(\mathsf{ALG})\,. \tag{5.5}$$

This follows from the observation that if the weight of $\mathsf{ALG}[\tau]$ was increased by some value $\zeta > 0$ at some step $t' < \tau$, then $\zeta$ also contributes to $\Delta^{t'}$Weights, so such contributions cancel out in (5.5). (There may be several such $\zeta$'s, as the weight of a packet may have been increased multiple times.)

Hence, using these bounds, as well as (5.3) or (5.4) for each event, yields $w^0(\mathsf{OPT}) = \sum_t \mathsf{advgain}^t \leq \sum_t \phi[\, w^t(\mathsf{ALG}[t]) - \Delta^t\text{Weights}\,] + \sum_\sigma (\Psi_{\sigma+1} - \Psi_\sigma) \leq \phi\, w^0(\mathsf{ALG})$, concluding the proof.

### 5.4 Arrival of a Packet

Let $\sigma$ be the index of the arrival event of a packet $j$ at the current time $t$. Let $P = P_\sigma$ be the plan just before $j$ arrives and let $Q = P_{\sigma+1}$ be the plan just after $j$ arrives. Our aim is to maintain invariants (InvA) and (InvP) using appropriate modifications of sets $\mathsf{ADV}$ and $\mathcal{F}$. We also show that the packet-arrival inequality (5.3) holds for $\sigma$. The algorithm does not change the weights and deadlines after packet arrival, so we will omit the superscript $t$ in the notation for weights and deadlines, that is $w_q = w_q^t$ and $d_q = d_q^t$, for each packet $q$. There are two cases, depending on whether or not $j \in Q$.

Case A.1: $j$ is not added to the plan, i.e., $Q = P$. This implies that $w_j < \mathsf{minwt}(P, d_j) = \mathsf{minwt}(Q, d_j)$. If $j \notin \mathsf{OPT}$, we do nothing. If $j \in \mathsf{OPT}$, we add a new shadow packet $s$ of weight $w_j$ to the adversary schedule $\mathsf{ADV}$ to the slot $\tau_j$ where $j$ is in $\mathsf{OPT}$. In both subcases the packet-arrival inequality (5.3) is trivial (as none of the sets involved in the potential change). Functions $\mathsf{pslack}()$ and $\#\mathsf{pairs}()$ do not change, so invariant (InvP) is preserved. Invariant (InvA) is preserved, since we either do not change $\mathsf{ADV}$ or we add a shadow packet $s$ in the slot $\tau_j$ for which $w_s \leq \mathsf{minwt}(P, \tau_j)$.

Case A.2: $j$ is added to the plan. Let $u$ be the lightest packet in $P$ with $d_u \leq \mathsf{nextts}(P, d_j)$; by assumption (UA1) such $u$ exists. As mentioned in Section 3, we have $Q = P \cup \{j\} \setminus \{u\}$ and $w_j > w_u$.

Replacing $u$ by $j$ in the plan can also trigger changes in $\mathcal{F}$, in cases when $u$ is in $\mathsf{ADV}$ or if $j$ is in $\mathsf{OPT}$. We divide the argument into two parts: (i) first we show that if $u \in \mathsf{ADV}$ then we can remove it, preserving the invariants and not decreasing the potential, and then (ii) assuming that $u \notin \mathsf{ADV}$, we analyze the effect of the remaining changes.

*Dealing with $u \in$ ADV.* If $u \in \mathsf{ADV}$, then we need to remove it from $\mathsf{ADV}$ to satisfy invariant (InvA) as $u \notin Q$. We replace $u$ in $\mathsf{ADV}$ by a new shadow packet $s$ of weight $w_u$, which is placed in $\mathsf{ADV}$ in the former slot $\tau_u$ of $u$, and we remove packet $F(u)$ from $\mathcal{F}$ (note that $F(u) \in \mathcal{F}$ is defined because $u \in \mathsf{ADV} \cap P$). The choice of $u$ implies that $w_s = w_u \leq \mathsf{minwt}(Q, \tau_u)$, thus preserving invariant (InvA). Using Lemma 5.3 for $u$ and $F(u)$ we get that invariant (InvP) is also preserved. As $w(\mathcal{F})$ decreases by $w_{F(u)}$ and $w(\mathsf{ADV} \cap P)$ decreases by $w_u$, the contribution of these changes to the potential change is $\frac{1}{\phi}(-w_{F(u)} + w_u) > 0$, by Lemma 5.1(c). Below, when bounding $\Psi_{\sigma+1} - \Psi_\sigma$, we will account for this contribution without an explicit reference.

*Analysis of other changes.* We can now proceed with the assumption that $u \notin \mathsf{ADV}$. There are several cases, depending on whether or not $j \in \mathsf{OPT}$ and on the ordering of $d_u$ and $d_j$.

Case A.2.a: $j \notin \mathsf{OPT}$. We do not further change $\mathsf{ADV}$, so invariant (InvA) holds. We have two sub-cases.

- Case A.2.a.P: $d_u \leq d_j$. We do not further change $\mathcal{F}$ or $\mathsf{ADV}$. Thus $\Psi_{\sigma+1} - \Psi_\sigma \geq \frac{1}{\phi}(w(Q) - w(P)) = \frac{1}{\phi}(w_j - w_u) > 0$. The function $\#\mathsf{pairs}()$ does not change and $\mathsf{pslack}()$ does not decrease, so invariant (InvP) is preserved.

- Case A.2.a.N: $d_u > d_j$. By the case assumption and the definition of $u$, both $d_j$ and $d_u$ are in the same segment of $P$. As $Q = P \cup \{j\} \setminus \{u\}$, we get that $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ for $\tau \in [d_j, d_u)$, while for other slots $\mathsf{pslack}()$ is not changed.

  We consider two further sub-cases. Let $\delta = \mathsf{prevts}(P, d_u)$. If there is no packet $f \in \mathcal{F}$ with $d_f \in (\delta, d_u)$, then we do nothing. In this case, inequality (5.3) holds trivially as $\Psi_{\sigma+1} - \Psi_\sigma \geq \frac{1}{\phi}(w_j - w_u) > 0$. Invariant (InvP) is preserved, since no pair changes and since for any $\tau \in [d_j, d_u) \subseteq (\delta, d_u)$ we have $\#\mathsf{pairs}(\tau) \leq 0$, whereas $\mathsf{pslack}(\tau)$ does not change for other $\tau$.

  The other sub-case is when there is $f \in \mathcal{F}$ with $d_f \in (\delta, d_u)$. Then let $f^* \in \mathcal{F}$ be the earliest-deadline packet with $d_{f^*} \in (\delta, d_u)$. We remove $f^*$ from $\mathcal{F}$ and add $u$ to $\mathcal{F}$. As $f^*$ is pending but not in $P$ and $d_{f^*} > \delta$, we get that $w_{f^*} < \mathsf{minwt}(P, d_u) \leq w_u$. Since also $w_u < w_j$, we obtain that $\Psi_{\sigma+1} - \Psi_\sigma \geq \frac{1}{\phi}(w_j - w_u + w_u - w_{f^*}) > 0$. This shows inequality (5.3). Also, $\#\mathsf{pairs}(\tau)$ decreases by 1 for $\tau \in [d_{f^*}, d_u)$, and $\#\mathsf{pairs}(\tau) \leq 0$ for $\tau \in (\delta, d_{f^*})$, even after replacing $f^*$ by $u$ in $\mathcal{F}$, showing that invariant (InvP) holds as well.

Case A.2.b: $j \in \mathsf{OPT}$. We add $j$ to $\mathsf{ADV}$ in the same slot as in $\mathsf{OPT}$, preserving invariant (InvA), and add $u$ to $\mathcal{F}$. We first analyze $\Psi_{\sigma+1} - \Psi_\sigma$. The weight of the plan increases by $w(Q) - w(P) = w_j - w_u$, the term $w(\mathsf{ADV} \cap P)$ increases by $w_j$, and $w(\mathcal{F})$ increases by $w_u$. Summing it up, $\Psi_{\sigma+1} - \Psi_\sigma \geq \frac{1}{\phi}((w_j - w_u) + w_u - w_j) = 0$. Hence the packet-arrival inequality (5.3) holds.

We now show that (InvP) continues to hold, splitting the proof into two cases:

- Case A.2.b.P: $d_u \leq d_j$ (the positive case). In this case, $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) + 1$ for $\tau \in [d_u, d_j)$, while for other slots $\mathsf{pslack}()$ is not changed. As $\#\mathsf{pairs}(\tau)$ increases by 1 for $\tau \in [d_u, d_j)$ and for other slots it stays the same, invariant (InvP) holds.

- Case A.2.b.N: $d_u > d_j$ (the negative case). We have that $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ for $\tau \in [d_j, d_u)$, while for other slots $\mathsf{pslack}()$ is not changed. As $\#\mathsf{pairs}(\tau)$ decreases by 1 for $\tau \in [d_j, d_u)$ and for other slots it stays the same, invariant (InvP) holds.

**5.5 Scheduling a Packet** After all packets with release time equal to $t$ arrive, the algorithm schedules its packet $p = \mathsf{ALG}[t]$. Let $j = \mathsf{ADV}[t]$ be the packet scheduled in $\mathsf{ADV}$ at time $t$. Recall that $j$ is *not* necessarily equal to $\mathsf{OPT}[t]$, the packet scheduled in $\mathsf{OPT}$ at time $t$; as a result of our modifications to $\mathsf{ADV}$, $j$ might be either a real packet that replaced $\mathsf{OPT}[t]$ or a shadow packet. Let $P = P^t$ be the plan just before scheduling $p$ and let $Q$ be the plan after the algorithm schedules $p$, possibly adjusts weights and deadlines, and after the time is incremented to $t+1$.

We split the analysis of the scheduling step into two parts, called the adversary step and the algorithm's step, defined as follows:

*Adversary step:* In the adversary step, the adversary schedules $j$, which is removed from $\mathsf{ADV}$, but the plan $P$ remains the same. Removing $j$ from $\mathsf{ADV}$ could trigger a change in $\mathcal{F}$. We show that these changes preserve both invariants (InvA) and (InvP) and we derive a bound (inequality (5.6)) on the change of the potential resulting from these changes. The analysis for this step is given in Section 5.5.1.

*Algorithm's step:* In the algorithm's step, the algorithm schedules $p$, the time is incremented to $t+1$, and the plan changes from $P$ to $Q$. The analysis of this step assumes that the changes described in the adversary step have already been implemented. (In particular, $j$ is removed from $\mathsf{ADV}$.) Using the bound (5.6), invariants (InvA) and (InvP), and other properties, we then show that the packet-scheduling inequality (5.4) holds after the sets $P$, $\mathsf{ADV}$, and $\mathcal{F}$ are updated to reflect the changes triggered by the scheduling step. We also show that invariants (InvA) and (InvP) are preserved.

The analysis of the algorithm's step is given in Sections 5.5.2-5.5.6. We first analyze the greedy step in Section 5.5.2. We then give a roadmap for the analysis of the leap step in Section 5.5.3, followed by the details of the analysis in Section 5.5.4, which describes the changes in $S_1$, and Sections 5.5.5-5.5.6 which contain the analysis of other changes resulting from a leap step.

**5.5.1 Adversary Step** The adversary schedules $j = \mathsf{ADV}[t]$, thus $j$ is removed from the adversary schedule $\mathsf{ADV}$. (Then $\mathsf{advgain}^t$ is the sum of $w_j^t$ and weight adjustments in $\mathsf{ADV}$, but we will not be dealing with $\mathsf{advgain}^t$ right now.) As we will not make other changes to $\mathsf{ADV}$, invariant (InvA) will be preserved. If $j \in P$, removing $j$ from $\mathsf{ADV}$ will also force us to remove a packet from $\mathcal{F}$. Here, we show that with appropriate changes invariant (InvP) will be preserved after the adversary step. Also, denoting by $\Delta_{\mathsf{ADV}}\Psi$ the change of the potential in the adversary step, we prove the following auxiliary inequality:

$$\Delta_{\mathsf{ADV}}\Psi - w_j^t \;\geq\; -\tfrac{1}{\phi^2}\, w_p^t - \tfrac{1}{\phi}\, w(\mathsf{sub}^t(p))\,. \tag{5.6}$$

The proof is divided into two cases, depending on whether or not $j \in P$. As packet weights are not changed in the adversary step, below we omit the superscript $t$ in the notations for weights.

<u>Case ADV.1</u>: $j \in P$. As $j \in \mathsf{ADV} \cap P$, packet $F(j) \in \mathcal{F}$ is defined. We remove $F(j)$ from $\mathcal{F}$. By Lemma 5.3, invariant (InvP) is preserved. Removing $j$ from $\mathsf{ADV}$ and $F(j)$ from $\mathcal{F}$ changes the potential by $\frac{1}{\phi}(-w_{F(j)} + w_j)$. By Lemma 5.1(b) we have $w(\mathsf{sub}^t(j)) \geq w_{F(j)}$. It follows that

$$\begin{aligned}\phi\,(\,\Delta_{\mathsf{ADV}}\Psi - w_j\,) &= -w_{F(j)} + w_j - \phi\, w_j\\ &= -\tfrac{1}{\phi}\, w_j - w_{F(j)}\\ &\geq -\tfrac{1}{\phi}\, w_j - w(\mathsf{sub}^t(j))\\ &\geq -\tfrac{1}{\phi}\, w_p - w(\mathsf{sub}^t(p))\,,\end{aligned}$$

where the last inequality follows from the choice of $p$ in line 1 of the algorithm's description; here we use that $j \in P$. This implies (5.6).

<u>Case ADV.2</u>: $j \notin P$. In this case we do not change $\mathcal{F}$, so invariant (InvP) is preserved. By invariant (InvA), $j$ is a shadow packet that satisfies $w_j \leq \mathsf{minwt}^t(t) = w_\omega$ as $\omega$ is the lightest packet in the first segment. Note that $w(\mathsf{sub}^t(\omega)) = w_\omega$ and that $\Delta_{\mathsf{ADV}}\Psi = 0$. Then

$$\begin{aligned}\phi\,(\,\Delta_{\mathsf{ADV}}\Psi - w_j\,) = -\phi\, w_j &\geq -\phi\, w_\omega\\ &= -\tfrac{1}{\phi}\, w_\omega - w(\mathsf{sub}^t(\omega))\\ &\geq -\tfrac{1}{\phi}\, w_p - w(\mathsf{sub}^t(p))\,,\end{aligned}$$

where the last inequality holds by the choice of $p$ again. This completes the proof of (5.6).

**5.5.2 Greedy Step** Recall that in a greedy step, the algorithm makes no changes in packet weights and deadlines; therefore, to simplify notation, for any packet $q$ we will write $w_q = w_q^t$ and $d_q = d_q^t$, omitting the superscript $t$. Let $\beta = \mathsf{nextts}^t(t)$ be the first tight slot in $P$, that is $S_1 = [t, \beta]$.

We start with some simple observations. According to the algorithm, $p$ is the heaviest packet in $S_1$. The algorithm does not adjust weights, so $\Delta^t\mathrm{Weights} = 0$. Since $\mathsf{sub}^t(p) = \omega$, inequality (5.6) gives us that $\Delta_{\mathsf{ADV}}\Psi - w_j \geq -w_p/\phi^2 - w_\omega/\phi$. As mentioned in Section 3, the new plan $Q$ (starting at time slot $t+1$) is $Q = P \setminus \{p\}$; thus the change of the potential associated with removing $p$ is $-w_p/\phi$.

We have two cases, depending on whether or not there is a packet in $\mathsf{ADV} \cap P$ in the first segment.

<u>Case G.1</u>: There is no packet in $\mathsf{ADV} \cap P$ with deadline in the first segment $S_1$. In this case, $p \notin \mathsf{ADV}$ (as $d_p \in S_1$) and we do not further change sets $\mathsf{ADV}$ and $\mathcal{F}$. So invariant (InvA) is preserved and $\mathsf{advgain}^t = w_j$. Observe that there is no packet $f \in \mathcal{F}$ with deadline in $S_1$; indeed, if such $f$ existed then packet $F^{-1}(f) \in \mathsf{ADV} \cap P$ would have its deadline in $S_1$, by invariant (InvP), contradicting the case condition. It follows that there is no $f \in \mathcal{F}$ with $d_f = t$, which implies that no packet in $\mathcal{F}$ expires in this step.

Invariant (InvP) continues to hold, because $\#\mathsf{pairs}(\tau) \leq 0$ for $\tau \in [t+1, \beta]$, even after the step, and for $\tau \geq d_p$ the value of $\mathsf{pslack}(\tau)$ does not change.

The calculation showing the packet-scheduling inequality (5.4) is now quite simple, as we just need to take into account bound (5.6), the adversary gain $\mathsf{advgain}^t = w_j$, and the contribution $\Delta_p\Psi = -w_p/\phi$ of updating the plan:

$$
\begin{aligned}
&\phi\,[\,w^t(\mathsf{ALG}[t]) - \Delta^t\text{Weights}\,] + (\Psi_{\sigma+1} - \Psi_\sigma) - \mathsf{advgain}^t\\
&= \phi\,[\,w_p - 0\,] + [\,\Delta_p\Psi + \Delta_{\mathsf{ADV}}\Psi\,] - w_j\\
&= \phi\,w_p + \Delta_p\Psi + [\,\Delta_{\mathsf{ADV}}\Psi - w_j\,]\\
&\geq \phi\,w_p - \tfrac{1}{\phi}\,w_p + \left[-\tfrac{1}{\phi^2}\,w_p - \tfrac{1}{\phi}\,w_\omega\right]\\
&= \tfrac{1}{\phi}\,w_p - \tfrac{1}{\phi}\,w_\omega \geq 0\,,
\end{aligned}
$$

where we use inequality $w_p \geq w_\omega$ in the last step, which follows from the definition of $\omega$.

Case G.2: There is a packet in $\mathsf{ADV} \cap P$ in the first segment $S_1$ (possibly $p \in \mathsf{ADV}$).

*Changing sets ADV and $\mathcal{F}$.* Let $g^*$ be the latest-deadline packet in $\mathsf{ADV} \cap P$ such that $d_{g^*} \leq \beta$ (which is defined by the case condition). Let $f_1$ be the earliest-deadline packet in $\mathcal{F}$ (which exists, because $\mathcal{F} \neq \emptyset$ by the existence of $g^*$); possibly $d_{f_1} = t$, which means that in such a case, $f_1$ cannot be in $\mathcal{F}$ in the next step.

If $p \in \mathsf{ADV}$, let $g = p$; otherwise let $g = g^*$. We remove $f_1$ from $\mathcal{F}$ and we replace $g$ in $\mathsf{ADV}$ by a new shadow packet $s$ of weight $w_s = \mathsf{minwt}^t(d_p) = \omega$, which is added to the slot of $g$ in $\mathsf{ADV}$. Note that now (after removing $f_1$), by Lemma 5.2, all packets in $\mathcal{F}$ have deadlines strictly after $t$, so none of them expires in this step.

*Preserving the invariants.* We now have $p \notin \mathsf{ADV}$ and the new shadow packet $s$ is in a slot within the first segment $S_1$ of $P$ and it satisfies $w_s \leq \omega$, so invariant (InvA) is preserved.

We next show that invariant (InvP) holds for any slot $\tau \geq t+1$ after the step. The value of $\mathsf{pslack}(\tau)$ decreases by 1 for slots $\tau \in [t+1, d_p)$ and for other slots it is not changed. We analyze how the values of $\#\mathsf{pairs}(\tau)$ change. If $d_{f_1} < d_g$, then $\#\mathsf{pairs}(\tau)$ decreases by 1 for $\tau \in [d_{f_1}, d_g)$ and for other slots it remains the same. Otherwise, $d_{f_1} \geq d_g$ and $\#\mathsf{pairs}(\tau)$ increases by 1 for $\tau \in [d_g, d_{f_1})$, while for other slots it does not change.

From the definitions of $f_1$, $g^*$, and invariant (InvP), we have that $\#\mathsf{pairs}(\tau) \leq 0$ holds for $\tau \in [t+1, d_{f_1}) \cup [d_{g^*}, \beta]$, even after the step. If follows that we just need to show that invariant (InvP) holds for $\tau \in [d_{f_1}, d_{g^*})$ and we can assume that $d_{f_1} < d_{g^*}$. Thus, as $d_{g^*} \leq \beta$ and $d_p \leq \beta$, invariant (InvP) holds for slots outside $S_1$.

We only need to consider the case when either $\#\mathsf{pairs}(\tau)$ increases or $\mathsf{pslack}(\tau)$ decreases, because in other cases the inequality $\mathsf{pslack}(\tau) \geq \#\mathsf{pairs}(\tau)$ is preserved. As both of these quantities change by at most 1, it is sufficient to show that in these two cases either (i) both quantities change in the same direction, or (ii) $\#\mathsf{pairs}(\tau) \leq 0$ after the step.

The first case, when $\#\mathsf{pairs}(\tau)$ increases, is actually already covered. Indeed, if $\#\mathsf{pairs}(\tau)$ increases, then $d_g < d_{f_1}$ and $\tau \in [d_g, d_{f_1})$, thus $\#\mathsf{pairs}(\tau) \leq 0$ as shown above (even after the step).

The second case, when $\mathsf{pslack}(\tau)$ decreases, happens when $\tau \in [t+1, d_p)$. This, combined with $\tau \in [d_{f_1}, d_{g^*})$, implies that $\tau \in [d_{f_1}, \min(d_p, d_{g^*}))$. As $g \in \{g^*, p\}$, it holds that $\tau \in [d_{f_1}, d_g)$, so $\#\mathsf{pairs}(\tau)$ decreases as well. Therefore, invariant (InvP) holds after the greedy step.

*Deriving inequality (5.4).* Let $\Delta_{p,g,f_1}\Psi$ be the change of $\Psi$ caused by removing $p$ from the plan, removing $f_1$ from $\mathcal{F}$, and $g$ from $\mathsf{ADV} \cap P$, that is, $\Delta_{p,g,f_1}\Psi = (-w_p - w_{f_1} + w_g)/\phi$. The adversary gain is $\mathsf{advgain}^t = w_g - w_s + w_j = w_g - w_\omega + w_j$. Note that $w_{f_1} \leq w_\omega$ as $f_1 \notin P$ and that $w_g \leq w_p$ as $p$ is the heaviest packet in $S_1$ and $d_g \leq \beta$. We show the packet-scheduling inequality (5.4) by summing these changes and the adversary gain bounded in (5.6):

$$
\begin{aligned}
&\phi\,[\,w^t(\mathsf{ALG}[t]) - \Delta^t\text{Weights}\,] + (\Psi_{\sigma+1} - \Psi_\sigma) - \mathsf{advgain}^t\\
&= \phi\,[\,w_p - 0\,] + [\,\Delta_{p,g,f_1}\Psi + \Delta_{\mathsf{ADV}}\Psi\,]\\
&\quad - [\,w_g - w_\omega + w_j\,]\\
&= \phi\,w_p + \Delta_{p,g,f_1}\Psi - w_g + w_\omega + [\,\Delta_{\mathsf{ADV}}\Psi - w_j\,]\\
&\geq \phi\,w_p + \tfrac{1}{\phi}\,[\,-w_p - w_{f_1} + w_g\,] - w_g + w_\omega\\
&\quad + \left[-\tfrac{1}{\phi^2}\,w_p - \tfrac{1}{\phi}\,w_\omega\right]\\
&= \tfrac{1}{\phi}\,w_p - \tfrac{1}{\phi}\,w_{f_1} - \tfrac{1}{\phi^2}\,w_g + \tfrac{1}{\phi^2}\,w_\omega\\
&\geq \tfrac{1}{\phi}\,w_p - \tfrac{1}{\phi}\,w_\omega - \tfrac{1}{\phi^2}\,w_p + \tfrac{1}{\phi^2}\,w_\omega\\
&= \tfrac{1}{\phi^3}\,(\,w_p - w_\omega\,) \,\geq\, 0\,,
\end{aligned}
$$

where the penultimate inequality holds by $w_{f_1} \leq w_\omega$ and by $w_g \leq w_p$, and the last inequality uses $w_p \geq w_\omega$. This concludes the analysis of a greedy step.

**5.5.3 Leap Step: a Roadmap** We now analyze the leap step of the algorithm, when it schedules a packet $p$ from a segment of $P^t$ other than $S_1$. In this case some packet weights change, so we use notation $w_a^t$ and $w_a^{t+1}$ (or $w^t(a)$ and $w^{t+1}(a)$) for the weights of a packet $a$ before and after $p$ is scheduled, respectively. For deadlines (which may also change), we implicitly assume that $d_a = d_a^t$, and write $d_a^{t+1}$ for the deadline of packet $a$ after scheduling $p$.

Recall that the new plan (starting at time $t+1$) is $Q = P \setminus \{p, \omega\} \cup \{\varrho\}$, where $\varrho = \mathsf{sub}^t(p)$. All changes in the plan are within two intervals of the plan: the first segment $S_1$ and the interval $[\delta, \gamma)$, where $\delta = \mathsf{prevts}^t(d_p^t)$

and $\gamma = \mathsf{nextts}^t(d^t_\varrho)$. Namely, $\omega$ is removed in $S_1$ and $p$ is replaced by $\varrho$ in $[\delta, \gamma)$. Furthermore, PlanM increases $\varrho$'s weight to $\mu \stackrel{\text{def}}{=} \mathsf{minwt}(P, d^t_\varrho)$, and, if this is an iterated leap step (i.e., $k > 1$ in the algorithm), it then modifies weights and deadlines of some packets $h_i$.

These changes in the plan may reduce some values of $\mathsf{pslack}(\tau)$ and may involve changes in ADV or $\mathcal{F}$. For example, if $\varrho$ is in $\mathcal{F}$, it will have to be removed, because $\mathcal{F}$ contains only pending packets that are not in the plan. This may trigger additional adjustments in ADV or $\mathcal{F}$, in order to restore invariants (InvA) and (InvP) after the move.

We start with two simple useful bounds. First, using inequality (5.6) and the definition of $\varrho$, we have

$$\Delta_{\mathsf{ADV}}\Psi - w^t_j \geq -\tfrac{1}{\phi^2}\, w^t_p - \tfrac{1}{\phi}\, w^t_\varrho. \tag{5.7}$$

Also, for any $\tau \geq t$, we have

$$\tfrac{1}{\phi^2}\, w^t_p + \tfrac{1}{\phi}\, w^t_\varrho \geq \omega^t \geq \mathsf{minwt}^t(\tau)\,, \tag{5.8}$$

where the first inequality follows from the choice of $p$ in line 1 of the algorithm (specifically, because the algorithm chose $p$ over $\omega$), and the second one follows from $\omega^t = \mathsf{minwt}^t(t) \geq \mathsf{minwt}^t(\tau)$, that is the monotonicity of $\mathsf{minwt}()$ with respect to $\tau$.

We now introduce several quantities that we will use in our estimates and in the proof of the packet scheduling inequality (5.4):

$\Delta^t w_\varrho \stackrel{\text{def}}{=} \mu - w^t_\varrho$ : The increase of the weight of $\varrho$ in line 4 of the algorithm.

$\Delta_{p,\omega,\varrho} w(P)$ : The change of the weight of the plan resulting from removing $p$, removing $\omega$, and adding $\varrho$ (with modified weight). Thus:

$$\begin{aligned}\Delta_{p,\omega,\varrho} w(P) &= -w^t_p - w^t_\omega + \mu \\ &\geq -(1 + \tfrac{1}{\phi^2})\, w^t_p - \tfrac{1}{\phi}\, w^t_\varrho + \mu\,,\end{aligned} \tag{5.9}$$

where the inequality follows from (5.8).

$\Delta^t w(H)$ : The total increase of the weights of packets $H = \{h_1, \ldots, h_k\}$ in line 11 in the algorithm. In case of a simple leap step (for $k = 0$), we set $H = \emptyset$ and $\Delta^t w(H) = 0$.

$\Delta^t\mathrm{Weights} = \Delta^t w_\varrho + \Delta^t w(H)$ : As already defined earlier, this is the total increase of the weights in step $t$.

$\Delta_{S_1}\Psi$ : The change of the potential due to modifications in ADV and $\mathcal{F}$ triggered by (but not including) the removal of $\omega$ from $S_1$. (See below for more details.)

$\Delta_{(\delta,\gamma]}\Psi$ : The change of the potential due to modifications in ADV and $\mathcal{F}$ triggered by (but not including) the replacement of $p$ by $\varrho$ in $(\delta, \gamma]$.

$\mathsf{advgain}^t_{(\delta,\gamma]}$ : The credit for the adversary for replacing packets in ADV with deadlines in $(\delta, \gamma]$ by lighter packets; it is equal to the total decrease of packet weights in ADV.

**Two key inequalities.** We will derive the proof of the packet-scheduling inequality (5.4) from the two key inequalities below, that bound the changes of the potential in intervals $S_1$ and $(\delta, \gamma]$:

$$\Delta_{S_1}\Psi \geq 0 \tag{5.10}$$

$$\begin{aligned}\Delta_{(\delta,\gamma]}\Psi - \phi\, \Delta^t w(H) - \mathsf{advgain}^t_{(\delta,\gamma]} \\ \geq -\tfrac{1}{\phi^2}\, w^t_p - \tfrac{1}{\phi}\, w^t_\varrho + \mu\end{aligned} \tag{5.11}$$

Note that the quantity $-\frac{1}{\phi^2}\, w^t_p - \frac{1}{\phi}\, w^t_\varrho + \mu$ is non-positive, by (5.8); thus if we change nothing while processing segments in $(\delta, \gamma]$ and if the weights of $h_i$'s do not change, (5.11) will hold.

**Deriving the packet-scheduling inequality.** Assuming that (5.10) and (5.11) hold, we now prove the packet-scheduling inequality (5.4). The total potential change in this step is

$$\Psi_{\sigma+1} - \Psi_\sigma = \Delta_{\mathsf{ADV}}\Psi + \tfrac{1}{\phi}\, \Delta_{p,\omega,\varrho} w(P) + \Delta_{S_1}\Psi + \Delta_{(\delta,\gamma]}\Psi\,,$$

because all changes in the plan and in sets ADV and $\mathcal{F}$ are accounted for (uniquely) in the terms on the right-hand side. (This will follow by examining changes detailed in Sections 5.5.1, 5.5.4 and 5.5.6). The total adversary gain is the sum of the gain from scheduling $j$ and the credits for decreasing weights in ADV, so

$$\mathsf{advgain}^t = w^t_j + \mathsf{advgain}^t_{(\delta,\gamma]}\,,$$

as the changes in $S_1$ will not involve any weight decreases for the adversary. Combining it all together, we have

$$\begin{aligned}
&\phi\,[\, w^t(\mathsf{ALG}[t]) - \Delta^t\mathrm{Weights}\,] + (\Psi_{\sigma+1} - \Psi_\sigma) - \mathsf{advgain}^t \\
&= \phi\,[\, w^t_p - \Delta^t w_\varrho - \Delta^t w(H)\,] \\
&\quad + [\, \Delta_{\mathsf{ADV}}\Psi + \tfrac{1}{\phi}\, \Delta_{p,\omega,\varrho} w(P) + \Delta_{S_1}\Psi + \Delta_{(\delta,\gamma]}\Psi\,] \\
&\quad - [\, w^t_j + \mathsf{advgain}^t_{(\delta,\gamma]}\,] \\
&= \phi\, w^t_p - \phi\,(\Delta^t w_\varrho) + \tfrac{1}{\phi}\, \Delta_{p,\omega,\varrho} w(P) + [\, \Delta_{\mathsf{ADV}}\Psi - w^t_j\,] \\
&\quad + \Delta_{S_1}\Psi + [\, \Delta_{(\delta,\gamma]}\Psi - \phi\, \Delta^t w(H) - \mathsf{advgain}^t_{(\delta,\gamma]}\,] \\
&\geq \phi\, w^t_p - \phi(\mu - w^t_\varrho) + \tfrac{1}{\phi}\, \Big[ -(1 + \tfrac{1}{\phi^2})\, w^t_p - \tfrac{1}{\phi}\, w^t_\varrho + \mu \Big] \\
&\quad + \Big[ -\tfrac{1}{\phi^2}\, w^t_p - \tfrac{1}{\phi}\, w^t_\varrho \Big] + 0 + \Big[ -\tfrac{1}{\phi^2}\, w^t_p - \tfrac{1}{\phi}\, w^t_\varrho + \mu \Big] \\
&= \Big( \phi - \tfrac{1}{\phi} - \tfrac{1}{\phi^3} - \tfrac{1}{\phi^2} - \tfrac{1}{\phi^2} \Big)\, w^t_p \\
&\quad + \Big( \phi - \tfrac{1}{\phi^2} - \tfrac{1}{\phi} - \tfrac{1}{\phi} \Big)\, w^t_\varrho + \Big( -\phi + \tfrac{1}{\phi} + 1 \Big)\, \mu \\
&= 0\,,
\end{aligned}$$

where in the inequality in the third step we use, in this order, inequalities (5.9), (5.7), (5.10), and (5.11), and in the last step we repeatedly use the definition of $\phi$.

Therefore, to complete the analysis, it is now sufficient to show that the two key inequalities (5.10) and (5.11) hold, and that invariants (InvA) and (InvP) are preserved after the step. We divide the proof into several parts, with the two main parts being:

*Processing $S_1$:* In this part, described in Section 5.5.4 below, we assume that the changes described in the adversary step have already been implemented. We describe changes in $\mathsf{ADV}$ and $\mathcal{F}$ triggered by the removal of $\omega$ from the plan. We then prove inequality (5.10) and that these changes preserve invariants (InvA) and (InvP). More precisely, we show that the invariants hold (with respect to packets) in $S_1$ and that they are not violated outside $S_1$, namely that for any $\tau \notin S_1$, the value of $\#\mathsf{pairs}(\tau)$ does not increase or $\#\mathsf{pairs}(\tau) \leq 0$ after these changes.

*Processing interval $(\delta, \gamma]$:* In this part, we assume that the changes described in the adversary step and in the processing of $S_1$ have already been implemented. We describe changes in $\mathsf{ADV}$ and $\mathcal{F}$ triggered by the replacement of $p$ by $\varrho$ and (for an iterated leap step) by modifications of packets $h_i$. We prove inequality (5.11) and that these changes preserve invariants (InvA) and (InvP). The proof will be divided into two cases, depending on whether it is a simple or an iterated leap step (see Sections 5.5.5 and 5.5.6, respectively). The proof for an iterated leap step is further divided into a number of smaller steps.

**5.5.4 Leap Step: Processing $S_1$** The change of the potential reflecting the removal of $\omega$ from $P$ has already been accounted for in $\Delta_{p,\omega,\varrho} w(P)$. However, after removing $\omega$ from $P$ we may also need to make changes in sets $\mathsf{ADV}$ and $\mathcal{F}$, in order to preserve the invariants in $S_1$. We refer to this process as "processing $S_1$", even though we do not actually change the plan; in fact, some modifications may involve pending packets (not in the plan) with deadlines after $S_1$. As explained earlier in Section 5.5.3, we assume that the changes in sets $\mathsf{ADV}$ and $\mathcal{F}$ described in Section 5.5.1 have already been implemented.

*Dealing with the case $\omega \in \mathsf{ADV}$.* We now consider the case when $\omega \in \mathsf{ADV}$. Since $\omega \in \mathsf{ADV} \cap P$, packet $F(\omega)$ is defined. We remove $F(\omega)$ from $\mathcal{F}$ and replace $\omega$ in $\mathsf{ADV}$ by a shadow packet of the same weight $w^t_\omega$, which is placed in the same time slot. This preserves invariant (InvA) in $S_1$. Removing $\omega$ from $\mathsf{ADV} \cap P$ and $F(\omega)$ from $\mathcal{F}$ causes the potential to change by $\frac{1}{\phi}(w^t_\omega - w^t_{F(\omega)}) > 0$, where the inequality follows from Lemma 5.1(c). Since the contribution of this change to $\Delta_{S_1}\Psi$ is positive, we can ignore it. Also, by Lemma 5.3, these removals preserve invariant (InvP) in all segments.

*Maintaining invariant (InvP) in $S_1$.* Since the value of $\mathsf{pslack}^t(\tau)$ decreases by 1 for $\tau \in [t+1, d_\omega)$, we may need to decrease $\#\mathsf{pairs}(\tau)$ for such $\tau$, if $\#\mathsf{pairs}(\tau) > 0$. Denoting by $f_1$ the earliest-deadline packet in $\mathcal{F}$, we consider two cases.

If $d_{f_1} \geq d_\omega$, then we do not make any further changes. Inequality 5.10 holds trivially. Since in this case $\#\mathsf{pairs}(\tau) \leq 0$ for $\tau \in [t+1, d_\omega)$ (even after the step), invariant (InvP) is maintained.

Otherwise, we have $d_{f_1} < d_\omega$. In this case we replace $f_1$ by $\omega$ in $\mathcal{F}$, which changes the potential by $\frac{1}{\phi}(w^t_\omega - w^t_{f_1}) > 0$ (where the inequality follows from $f_1 \notin P$), implying (5.10). To show that (InvP) is preserved, note that after replacing $f_1$ by $\omega$ in $\mathcal{F}$, the value of $\#\mathsf{pairs}(\tau)$ decreases by 1 for $\tau \in [d_{f_1}, d_\omega)$ and for other slots it remains the same; in particular, we have $\#\mathsf{pairs}(\tau) \leq 0$ for $\tau \in [t+1, d_{f_1})$. Hence, invariant (InvP) is preserved after processing $S_1$.

*No packet from $\mathcal{F}$ expires.* We claim that after processing $S_1$, there is no packet in $\mathcal{F}$ that expires in the current step. This holds by Lemma 5.2 if $d_{f_1} < d_\omega$, as in this case we removed $f_1$ from $\mathcal{F}$. Consider the other case, when $d_{f_1} \geq d_\omega$. Then the claim trivially holds if $d_{f_1} > t$. Suppose for a contradiction that $d_{f_1} = d_\omega = t$. Thus $S_1$ consists of just a single slot $t$; in other words, $\mathsf{pslack}(P, t) = 0$. Using invariant (InvP) we get that packet $g_1 = F^{-1}(f_1) \in \mathsf{ADV} \cap P$ also satisfies $d_{g_1} = t$. This leads to a contradiction, because after the adversary step there is no packet in $\mathsf{ADV} \cap P$ with deadline equal to $t$ (see Section 5.5.1).

**5.5.5 Processing $(\delta, \gamma]$ in a Simple Leap Step (Case L1)** We now analyze the effects of replacing $p$ by $\varrho$ in $P$, in the case of a simple leap step, namely when $k = 0$ in the algorithm. Recall that the contribution of this change in the plan to the potential is already accounted for in $\Delta_{p,\omega,\varrho} w(P)$, but these changes may trigger modifications in $\mathsf{ADV}$ and $\mathcal{F}$, in order to restore the invariants. We assume that the changes in sets $\mathsf{ADV}$ and $\mathcal{F}$ described in Section 5.5.1 and in Section 5.5.4 have already been implemented. (We note that these changes might have involved some packets considered in this section; for example $\varrho$ might have been removed from $\mathcal{F}$ when processing $S_1$, if we earlier had $\varrho = F(\omega)$.)

In the simple leap step $d_\varrho$ and $d_p$ are in the same segment $(\delta, \gamma]$, that is $\mathsf{nextts}^t(d_\varrho) = \mathsf{nextts}^t(d_p) = \gamma$. We have $H = \emptyset$ and $\Delta^t w(H) = 0$. There are two subcases, depending on whether some changes are needed or not.

<u>Case L.1.A</u>: $\varrho \notin \mathcal{F}$ and there is no packet in $\mathsf{ADV} \cap P$ with deadline in $(\delta, \gamma]$; in particular $p \notin \mathsf{ADV} \cap P$. Then

we do not further change the set $\mathcal{F}$ or $\mathsf{ADV}$. We have $\Delta_{(\delta,\gamma]}\Psi = 0$, $\mathsf{advgain}^t_{(\delta,\gamma]} = 0$, and the left-hand side of (5.11) is zero. As the right-hand side is non-positive, (5.11) holds.

Invariant (InvP) implies that $\#\mathsf{pairs}(\gamma) \le 0$; so using the case assumption we get that $\#\mathsf{pairs}(\tau) \le 0$ for all $\tau \in (\delta,\gamma]$, implying that invariant (InvP) is preserved after the step. As $\varrho \notin \mathsf{ADV}$ and $p \notin \mathsf{ADV}$, invariant (InvA) holds as well.

Case L.1.B: $\varrho \in \mathcal{F}$ or there is a packet in $\mathsf{ADV} \cap P$ with deadline in $(\delta,\gamma]$. In this case, $\mathsf{ADV}$ and $\mathcal{F}$ will be changed to maintain invariants (InvP) and (InvA).

*Changes in Case L.1.B.* Let $g^*$ be the latest-deadline packet in $\mathsf{ADV}\cap P$ with $d_{g^*} \le \gamma$. We note that $g^*$ is well defined. This is trivially true if the second condition of the case is satisfied. If $\varrho \in \mathcal{F}$ then $F^{-1}(\varrho)$ is a candidate, because $d_\varrho \le \gamma$, and thus $d_{F^{-1}(\varrho)} \le \gamma$ as well, by invariant (InvP). (It is possible that $d_{g^*} \le \delta$ in this case.)

Similarly, let $f^*$ be the earliest-deadline packet in $\mathcal{F}$ with $d_{f^*} > \delta$. This $f^*$ is also well-defined, because either $\varrho \in \mathcal{F}$, in which case $\varrho$ is a candidate, or $d_{g^*} \in (\delta,\gamma]$, in which case $F(g^*)$ is a candidate by Lemma 5.1(a). (It is possible that $d_{f^*} > \gamma$.)

We now define packets $g$ and $f$, and we modify $\mathcal{F}$ and $\mathsf{ADV}$ as follows. If $p \in \mathsf{ADV}$, let $g = p$; otherwise let $g = g^*$. If $\varrho \in \mathcal{F}$, let $f = \varrho$; otherwise let $f = f^*$. We remove $f$ from $\mathcal{F}$ and we replace $g$ in $\mathsf{ADV}$ by a new shadow packet $s$ of weight $\mu = \mathsf{minwt}^t(d_p)$, added to the slot of $g$ in $\mathsf{ADV}$. It follows that $g$ is no longer in $\mathsf{ADV}\cap P$.

*Calculation in Case L.1.B.* Note that $w^t_f \le w^t_\varrho$ as $d_f > \delta$ and as $\varrho$ is the heaviest pending packet not in $P$ with deadline after $\delta$. Furthermore, $w^t_g \le w^t_p$ as $w(\mathsf{sub}(P,g)) \ge w^t_\varrho$ and thus if $w^t_g > w^t_p$, the algorithm would schedule $g$ instead of $p$. Thus the changes described above give us that

$$\begin{aligned}\Delta_{(\delta,\gamma]}\Psi - \mathsf{advgain}^t_{(\delta,\gamma]} &= \tfrac{1}{\phi}\,(-w^t_f + w^t_g) - (w^t_g - \mu)\\ &= -\tfrac{1}{\phi}\,w^t_f - \tfrac{1}{\phi^2}\,w^t_g + \mu\\ &\ge -\tfrac{1}{\phi}\,w^t_\varrho - \tfrac{1}{\phi^2}\,w^t_p + \mu\,,\end{aligned}$$

which shows (5.11).

*Invariants in Case L.1.B.* Since after the changes it holds $\varrho \notin \mathsf{ADV}$ and $p \notin \mathsf{ADV}$ and since $w_s = \mathsf{minwt}^t(d_p)$, invariant (InvA) is maintained. We now show that invariant (InvP) is preserved after we remove $f$ from $\mathcal{F}$ and $g$ from $\mathsf{ADV}$.

The value of $\mathsf{pslack}(\tau)$ increases by 1 for $\tau \in [d_p, d_\varrho)$ if $d_p < d_\varrho$, and decreases by 1 for $\tau \in [d_\varrho, d_p)$ if $d_\varrho < d_p$, while for other slots in $(\delta,\gamma]$ it remains the same. If $d_f < d_g$, then $\#\mathsf{pairs}(\tau)$ decreases by 1 for $\tau \in [d_f, d_g)$. Otherwise, $d_f \ge d_g$ and $\#\mathsf{pairs}(\tau)$ increases by 1 for $\tau \in [d_g, d_f)$. For other slots, $\#\mathsf{pairs}(\tau)$ remains the same.

From the definitions of $f^*$, $g^*$, and invariant (InvP), we have that $\#\mathsf{pairs}(\tau) \le 0$ holds for $\tau \in (\delta, d_{f^*}) \cup [d_{g^*}, \gamma]$, even after the step. Thus (InvP) holds in this range, which includes slots outside $(\delta,\gamma]$ if $d_{f^*} > \gamma$ or if $d_{g^*} \le \delta$ (note that either of the two conditions implies $d_{f^*} > d_{g^*}$). So for the rest of the proof we can assume that $\delta < d_{f^*} < d_{g^*} \le \gamma$ and that $\tau \notin (\delta, d_{f^*}) \cup [d_{g^*}, \gamma]$.

Moreover, as $d_{f^*}, d_{g^*} \in (\delta,\gamma]$ (by the assumption above) and $d_\varrho, d_p \in (\delta,\gamma]$, we have that the values of $\mathsf{pslack}(\tau)$ and $\#\mathsf{pairs}(\tau)$ remain unchanged for all slots $\tau \notin (\delta,\gamma]$, preserving invariant (InvP) for these slots.

Thus now it only remains to show that invariant (InvP) is preserved for slots $\tau \in [d_{f^*}, d_{g^*}) \subset (\delta,\gamma]$. Further, since the values of $\#\mathsf{pairs}(\tau)$ and $\mathsf{pslack}(\tau)$ change by at most 1, it is sufficient to show that each $\tau \in [d_{f^*}, d_{g^*})$ satisfies the following two conditions: (i) if $\#\mathsf{pairs}(\tau)$ increases then so does $\mathsf{pslack}(\tau)$, and (ii) if $\mathsf{pslack}(\tau)$ decreases then so does $\#\mathsf{pairs}(\tau)$.

To show (i), suppose that $\#\mathsf{pairs}(\tau)$ increases. This happens only when $d_g < d_f$ and $\tau \in [d_g, d_f)$. Since also $\tau \in [d_{f^*}, d_{g^*})$, we get that $g \ne g^*$ and $f \ne f^*$. Therefore $g = p$ and $f = \varrho$, which means that $\mathsf{pslack}(\tau)$ also increases.

To show (ii), suppose that $\mathsf{pslack}(\tau)$ decreases, which happens only when $d_\varrho < d_p$ and $\tau \in [d_\varrho, d_p)$. This, combined with $\tau \in [d_{f^*}, d_{g^*})$, implies that $\tau \in [\max(d_\varrho, d_{f^*}), \min(d_p, d_{g^*}))$. As $f \in \{f^*, \varrho\}$ and $g \in \{g^*, p\}$, it follows that $\tau \in [d_f, d_g)$, so $\#\mathsf{pairs}(\tau)$ decreases as well.

**5.5.6 Processing $(\delta,\gamma]$ in an Iterated Leap Step (Case L2)** Here we address the last (and most involved) part of our argument, that is the analysis of an iterated leap step, namely when $k \ge 1$ in the algorithm. As in the previous section, we assume that $\mathsf{ADV}$ and $\mathcal{F}$ have already been modified, as described in Sections 5.5.1 (the adversary step) and 5.5.4 (processing $S_1$). We now need to estimate the potential change due to the changes triggered by the replacement of $p$ by $\varrho$ and by the "shifting" of $h_i$'s, prove key inequality (5.11), and that invariants (InvA) and (InvP) hold after the step.

As before, $\delta = \mathsf{prevts}^t(d_p)$ and $\gamma = \mathsf{nextts}^t(d_\varrho)$. Recall that in an iterated leap step we have $d_\varrho > \mathsf{nextts}^t(d_p)$, so the interval $(\delta,\gamma]$ is a union of two or more consecutive segments of $P$. Let $h_0 = p, h_1, \ldots, h_k$ be the packets from Algorithm $\mathsf{PlanM}$ (line 9) and let $h_{k+1} = \varrho$. All packets $h_i$, $i = 0, \ldots, k$, are in different segments of $P$, not necessarily consecutive. As in the algorithm, let $\tau_i = \mathsf{nextts}^t(d^t_{h_i}), i = 0, \ldots, k$; note that $\tau_k = \gamma$. To simplify notation, let $\mu_i = \mathsf{minwt}^t(\tau_i)$; we have $\mu_k = \mathsf{minwt}^t(d^t_\varrho) = \mu$.

We now prove a useful bound of the increase of the weights of a subset of packets $h_i$.

LEMMA 5.4. *For any $a', b'$ satisfying $1 \le a' \le b' \le k$, let $\Delta^t w(h_{a'}, \ldots, h_{b'})$ be the total amount by which the algorithm increases the weights of packets $h_{a'}, \ldots, h_{b'}$. Suppose that there exists $i \in [a', b']$ such that $w^t_{h_i} < \mu_{i-1}$, i.e., the algorithm increases the weight of $h_i$. Then $\Delta^t w(h_{a'}, \ldots, h_{b'}) \le \mu_{a'-1} - w^t_{h_{b'}}$.*

*Proof.* Let $c \in [a', b']$ be the maximum index such that $w^t_{h_c} < \mu_{c-1}$; such $c$ exists by the assumption of the lemma. We show the claim as follows:

$$\begin{aligned}
\Delta^t w(h_{a'}, \ldots, h_{b'}) &= \sum_{i=a'}^{c} (\max(\mu_{i-1}, w^t_{h_i}) - w^t_{h_i}) \\
&= \sum_{i=a'}^{c} \max(\mu_{i-1} - w^t_{h_i}, 0) \\
&\le \sum_{i=a'}^{c-1} \max(\mu_{i-1} - \mu_i, 0) \qquad (5.12)\\
&\quad + \max(\mu_{c-1} - w^t_{h_c}, 0) \\
&= \sum_{i=a'}^{c-1} (\mu_{i-1} - \mu_i) + \mu_{c-1} - w^t_{h_c} \qquad (5.13)\\
&= \mu_{a'-1} - w^t_{h_c} \\
&\le \mu_{a'-1} - w^t_{h_{b'}} . \qquad (5.14)
\end{aligned}$$

where inequality (5.12) follows from $w^t_{h_i} \ge \mu_i$, equality (5.13) from $\mu_{i-1} \ge \mu_i$ and from $\mu_{c-1} > w^t_{h_c}$ (by the choice of $c$), and inequality (5.14) from $w^t_{h_c} \ge w^t_{h_{b'}}$ as $c \le b'$.

Similarly as in Section 5.5.5 we have two cases.

Case L.2.A: $\varrho \notin \mathcal{F}$ and there is no packet in $\mathsf{ADV} \cap P$ with deadline in $(\delta, \gamma]$. Then we do not make any changes in $\mathsf{ADV}$ and $\mathcal{F}$. From the case condition, no $h_i$, for $i = 0, \ldots, k$, is in $\mathsf{ADV}$, because each $h_i$ is in $P$ and its deadline is in $(\delta, \gamma]$. According to invariant (InvP) we have that $\#\mathsf{pairs}(\gamma) \le 0$, and then the second part of the case condition implies that in fact $\#\mathsf{pairs}(\tau) \le 0$ for all $\tau \in (\delta, \gamma]$, implying that invariant (InvP) holds after the step. Also, invariant (InvA) continues to hold as $p \notin \mathsf{ADV}$ and we do not change $\mathsf{ADV}$.

It remains to show (5.11). Since we have not changed $\mathsf{ADV}$, we have $\mathsf{advgain}^t_{(\delta,\gamma]} = 0$. Next, we claim that $\Delta^t w(H) \le \mu_0 - \mu_k$. If there is no $i \in [1, k]$ such that $w^t_{h_i} < \mu_{i-1}$, then $\Delta^t w(H) = 0 \le \mu_0 - \mu_k$ as $\mu_0 \ge \mu_k$. Otherwise, we use Lemma 5.4 with $a' = 1$ and $b' = k$ to get $\Delta^t w(H) \le \mu_0 - w^t_{h_k} \le \mu_0 - \mu_k$, with the last inequality following from $w^t_{h_k} \ge \mu_k$.

In this case, the potential change $\Delta_{(\delta,\gamma]}\Psi$ only reflects the increase of the weights of $h_i$'s, since all $h_i$'s are in $Q$ and we do not make other changes in the plan (besides removing $p$ and $\omega$ and adding $\varrho$, which are already accounted for in $\Delta_{p,\omega,\varrho} w(P)$). Then (5.11) follows from the above bound on $\Delta^t w(H)$ and an easy calculation:

$$\begin{aligned}
\Delta_{(\delta,\gamma]}\Psi - \phi\, \Delta^t w(H) - \mathsf{advgain}^t_{(\delta,\gamma]} &= \tfrac{1}{\phi}\, \Delta^t w(H) - \phi\, \Delta^t w(H) - 0 \\
&= -\Delta^t w(H) \\
&\ge -\mu_0 + \mu_k \\
&\ge -\tfrac{1}{\phi^2}\, w^t_p - \tfrac{1}{\phi}\, w^t_\varrho + \mu_k ,
\end{aligned}$$

where the last inequality follows from $\mu_0 \le \frac{1}{\phi^2}\, w^t_p + \frac{1}{\phi}\, w^t_\varrho$, which is (5.8) with $\tau = d_p$ (as $\mu_0 = \mathsf{minwt}^t(d_p)$).

Case L.2.B: $\varrho \in \mathcal{F}$ or there is a packet in $\mathsf{ADV} \cap P$ with deadline in $(\delta, \gamma]$. In this case sets $\mathsf{ADV}$ and $\mathcal{F}$ will be changed. We focus on the segments which contain the packets $h_0 = p, h_1, ..., h_k$ that are modified by the algorithm. Specifically, for $i = 0, \ldots, k$, let $S'_i$ be the segment of $P$ that ends at $\tau_i = \mathsf{nextts}^t(d^t_{h_i})$, that is the segment containing $d^t_{h_i}$. Recall that $\mathsf{prevts}^t(p) = \delta$, $\tau_k = \gamma$, and that we defined $h_{k+1} = \varrho$.

We start by defining a packet $g \in \mathsf{ADV} \cap P$. Let $g^*$ be the latest-deadline packet in $\mathsf{ADV} \cap P$ with $d_{g^*} \le \gamma$. Observe that packet $g^*$ is well defined. This is trivially true if the second part of the case condition holds. Otherwise, we have $\varrho \in \mathcal{F}$, in which case packet $F^{-1}(\varrho) \in \mathsf{ADV} \cap P$ is a candidate for $g^*$, because $\mathsf{prevts}^t(d_{F^{-1}(\varrho)}) < d_\varrho \le \gamma$, by Lemma 5.1(a). (It is possible that $d_{g^*} \le \delta$.) We now define $g$ as follows: If $d^t_{g^*}$ is in a segment $S'_i$ for some $i$ and $h_i \in \mathsf{ADV}$, then let $g = h_i$; otherwise, let $g = g^*$. Observe that if $h_k \in \mathsf{ADV}$, then $g = h_k$.

We will process segments $S'_i$ in *groups*, where each group is specified by some non-empty interval of indices $[a, b] \subseteq \{0, \ldots, k\}$ of segments $S'_i$. Roughly, we have a group for each $h_i \in \mathsf{ADV}$ (that needs to be replaced in $\mathsf{ADV}$ because its deadline was decreased), a special last group, and possibly a special group at the beginning. Let $i_1 < i_2 < \cdots < i_\ell$ be the indices of those packets $h_0 = p, h_1, ..., h_k$ that are in $\mathsf{ADV}$. Note that $d^t_g \in [d^t(h_{i_\ell}), \gamma]$, because $h_{i_\ell} \in \mathsf{ADV}$ is a candidate for $g^*$. In particular, since $g \in \mathsf{ADV}$, we have that $g \notin \{h_0, ..., h_k\} - \{h_{i_\ell}\}$; that is, among *all* packets $h_0, ..., h_k$, $g$ may be possibly equal only to $h_{i_\ell}$. The definition of these groups depends on whether $\ell > 0$ or $\ell = 0$ (that is, when none of packets $h_i$ is in $\mathsf{ADV}$):

Case $\ell > 0$: For each $a = 1, \ldots, \ell - 1$ (if $\ell > 1$), the interval $[i_a, i_{a+1}-1]$ is a *middle group.* If $i_1 > 0$, i.e., if $h_0 = p \notin \mathsf{ADV}$, then there is a special *initial group* $[0, i_1 - 1]$. This group does not exist if $i_1 = 0$. Next, we assign the indices in $[i_\ell, k]$ to one or two groups. If $g = h_{i_\ell}$ (in particular, if $i_\ell = k$), then $[i_\ell, k]$ is the *terminal group.* Otherwise, if $g \neq h_{i_\ell}$, let $\alpha$ be the smallest index in $0, ..., k$ for which $\tau_\alpha \geq d^t_g$. The assumption that $g \neq h_{i_\ell}$ implies that $\alpha > i_\ell$. Then $[\alpha, k]$ is the terminal group and $[i_\ell, \alpha - 1]$ is a new middle group. See Figure 3 for an illustration.

Case $\ell = 0$: We create at most two groups only. There is the terminal group $[\alpha, k]$, where $\alpha$ is again the smallest index in $0, ..., k$ with $\tau_\alpha \geq d^t_g$, and if $\alpha > 0$, we also have the initial group $[0, \alpha - 1]$ .

Note that in almost every group $[a, b]$ packet $h_a$ is in $\mathsf{ADV}$; the only two possible exceptions are (i) the initial group, and (ii) the terminal group in case when $\ell = 0$ or $g \neq h_{i_\ell}$. On the other hand, packets $h_{a+1}, \ldots, h_b$ are never in $\mathsf{ADV}$.

| | | $\delta$ $S'_0$ | | $S'_1$ | $S'_2$ | | $S'_3$ | | | $S'_4$ | $S'_5$ | | | $S'_6$ | | $S'_7$ | $\gamma$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P^t$ | $\cdots$ | $p = h_0$ | | $h_1$ | $h_2$ | | $h_3$ | | | $h_4$ | $h_5$ | | | $h_6$ | $g^*$ | $h_7$ | $\cdots$ |

Figure 3: In this example with $k = 7$, tight slots are depicted by vertical line segments and the circled packets are in $\mathsf{ADV}$. Thus $[0, 1]$ is the initial group, $[2, 2]$, $[3, 3]$, $[4, 5]$, and $[6, 6]$ are the middle groups, and $[7, 7]$ is the terminal group.

To show (5.11), we split the potential changes and the adversary credit for replacing packets in $\mathsf{ADV}$ among groups in a natural way. Namely, for a group $[a, b]$, let $\Delta_{[a,b]}\Psi$ be the total change of the potential due to changes done when processing group $[a, b]$, let $\mathsf{advgain}^t_{[a,b]}$ be the adversary credit for the changes of $\mathsf{ADV}$ when processing group $[a, b]$ (that is for replacing $h_a$ or $g$ by a lighter packet), and let $\Delta^t w(h_{a+1}, \ldots, h_{b+1})$ be the total amount by which the algorithm increases the weights of $h_{a+1}, \ldots, h_{b+1}$. Our goal is to prove that for each middle group $[a, b]$ and for the possible initial group $[a, b]$ (which has $a = 0$) it holds

$$\begin{aligned} \Delta_{[a,b]}\Psi - \phi\, \Delta^t w(h_{a+1}, \ldots, h_{b+1}) - \mathsf{advgain}^t_{[a,b]} \\ \geq -\tfrac{1}{\phi^2}\, w^t_{h_a} + \tfrac{1}{\phi^2}\, w^t_{h_{b+1}}\,. \end{aligned} \tag{5.15}$$

Similarly, for the terminal group $[a, k]$, which is defined in all cases, we show

$$\begin{aligned} \Delta_{[a,k]}\Psi - \phi\, \Delta^t w(h_{a+1}, \ldots, h_k) - \mathsf{advgain}^t_{[a,k]} \\ \geq -\tfrac{1}{\phi^2}\, w^t_{h_a} - \tfrac{1}{\phi}\, w^t_\varrho + \mu_k\,. \end{aligned} \tag{5.16}$$

(Note that the right-hand side of (5.16) may be positive.) The sum of (5.15) over all middle groups and the possible initial group plus (5.16) gives us exactly the key inequality (5.11). This is because all terms $\frac{1}{\phi^2}\, w^t_{h_{b+1}}$ on the right-hand side of inequality (5.15) will cancel as they appear in the inequality for the next group with a negative sign, so the right-hand sides of all the inequalities for all groups add up to $-\frac{1}{\phi^2}\, w^t_{h_0} - \frac{1}{\phi}\, w^t_\varrho + \mu_k = -\frac{1}{\phi^2}\, w^t_p - \frac{1}{\phi}\, w^t_\varrho + \mu$.

We process the groups in the reverse order of time, i.e., from the last one, which is always the terminal group, to the first one, which may be of any type. We maintain the property that after processing each group $[a, b]$ packet $h_a$ will not be in $\mathsf{ADV}$ (even though it may have been in $\mathsf{ADV}$ earlier).

Regarding invariant (InvP) the value of $\mathsf{pslack}()$ may change only for slots $\tau \in S_1 \cup S'_0 \cup \cdots \cup S'_k$. We have already shown how to maintain invariant (InvP) in $S_1$. For the remaining slots, we analyze the changes of $\mathsf{pslack}()$ and $\#\mathsf{pairs}()$ in $(\delta, \gamma]$ when we derive inequalities (5.15) and (5.16) for each group $[a, b]$ of segments. When processing a group $[a, b]$ we will show how to preserve invariant (InvP) for slots $\tau$ in segments $S'_a, \ldots, S'_b$, and that invariant (InvP) is not affected for slots $\tau$ outside these segments, that is for such $\tau$ we will have that either $\#\mathsf{pairs}(\tau)$ does not increase or $\#\mathsf{pairs}(\tau) \leq 0$ after processing the group. In a similar way we show that invariant (InvA) holds after the step.

**Processing the terminal group.** Let $[a, k]$ be the interval of indices representing the terminal group of segments.

Let $f^*$ be the earliest-deadline packet in $\mathcal{F}$ with $d_{f^*} > \delta$. The assumption of Case L.2.B implies that $f^*$ is well defined. Indeed, this is trivial if $\varrho \in \mathcal{F}$. If $\varrho \notin \mathcal{F}$ then there exists a packet $g' \in \mathsf{ADV} \cap P$ with $d_{g'} \in (\delta, \gamma]$ and then the packet $F(g')$ is a candidate for $f^*$, because $d_{F(g')} > \mathsf{prevts}^t(d^t_{g'}) \geq \delta$ by Lemma 5.1(a). (It may happen that $d_{f^*} > \gamma$.)

We now define a packet $f \in \mathcal{F}$ and modify sets $\mathsf{ADV}$ and $\mathcal{F}$ as follows. If $\varrho \in \mathcal{F}$, let $f = \varrho$; otherwise, let $f = f^*$. By the choice of $f \in \{\varrho, f^*\}$ and the definition of $\varrho$ we have that $w^t_f \leq w^t_\varrho$. We remove $f$ from $\mathcal{F}$ and in $\mathsf{ADV}$ we replace packet $g$ by a new shadow packet $s$ of weight $\mathsf{minwt}^t(d^t_g)$, placed in the same slot as $g$. This preserves invariant (InvA) in segments $S'_a, \ldots, S'_k$.

*Calculation showing* (5.16) *for the terminal group.* Apart from the changes in the paragraph above, we need to take into account the possible change of weights of packets $h_{a+1}, \ldots, h_k$, which also increases the weight of the plan, because all packets $h_1, ..., h_k$ remain in the new plan $Q$. (Increasing the weight of $\varrho = h_{k+1}$ has already been accounted for in $\Delta_{p,\omega,\varrho} w(P)$.)

We claim that $\Delta^t w(h_{a+1},\ldots,h_k) \le \mu_a - \mu_k$. There are two simple cases. If there is no $i \in [a+1,k]$ such that $w^t_{h_i} < \mu_{i-1}$, then $\Delta^t w(h_{a+1},\ldots,h_k) = 0 \le \mu_a - \mu_k$ as $\mu_a \ge \mu_k$. Otherwise, we use Lemma 5.4 with $a' = a+1$ and $b' = k$ to get $\Delta^t w(h_{a+1},\ldots,h_k) \le \mu_a - w^t_{h_k} \le \mu_a - \mu_k$, where the last inequality follows from $w^t_{h_k} \ge \mu_k$. The claim is thus proved.

The second claim is that in this case we have $w^t_g \le w^t_{h_a}$. This is trivial if $g = h_{i_\ell}$ and thus $a = i_\ell$. Otherwise, recall that, by the definition of the terminal group, $a = \alpha$ is the smallest index $\alpha$ with $\tau_\alpha \ge d^t_g$, that $a > i_\ell$, and that $h_a$ is the heaviest packet in plan $P$ with $d_{h_a} \in (\tau_{a-1},\gamma]$. As $g$ was in $\mathsf{ADV} \cap P$ and as $d^t_g \in (\tau_{a-1},\gamma]$ by the definition of $a = \alpha$, we get that $w^t_g \le w^t_{h_a}$.

Using the two claims shown above, we derive inequality (5.16) as follows:

$$\begin{aligned}
&\Delta_{[a,k]}\Psi - \phi\,\Delta^t w(h_{a+1},\ldots,h_k) - \mathsf{advgain}^t_{[a,k]}\\
&= \tfrac{1}{\phi}\left(\Delta^t w(h_{a+1},\ldots,h_k) - w^t_f + w^t_g\right)\\
&\quad - \phi\,\Delta^t w(h_{a+1},\ldots,h_k) - (w^t_g - \mathsf{minwt}^t(d^t_g))\\
&= -\tfrac{1}{\phi^2}\,w^t_g - \tfrac{1}{\phi}\,w^t_f - \Delta^t w(h_{a+1},\ldots,h_k) + \mathsf{minwt}^t(d^t_g)\\
&\ge -\tfrac{1}{\phi^2}\,w^t_{h_a} - \tfrac{1}{\phi}\,w^t_\varrho - (\mu_a - \mu_k) + \mu_a\\
&= -\tfrac{1}{\phi^2}\,w^t_{h_a} - \tfrac{1}{\phi}\,w^t_\varrho + \mu_k\,,
\end{aligned}$$

where the last inequality follows from the claims above, $w^t_f \le w^t_\varrho$ (as explained earlier), and $\mathsf{minwt}^t(d^t_g) \ge \mu_a$, that follows from $\tau_a \ge d^t_g$.

*Invariant (InvP) after processing the terminal group.* We claim that after we process the terminal group, invariant (InvP) holds for slots in segments $S'_a, S'_{a+1},\ldots,S'_k$ and invariant (InvP) is not affected for another slot. (The proof is similar to the one in Case L.1.B.)

Recall that the value of $\mathsf{pslack}(\tau)$ increases by 1 for $\tau \in [d^t_{h_i},\tau_i)$, $i = a,\ldots,k-1$. Moreover, if $d^t_{h_k} < d_\varrho$, the value of $\mathsf{pslack}(\tau)$ increases by 1 for $\tau \in [d^t_{h_k}, d_\varrho)$, and otherwise, it decreases by 1 for $\tau \in [d_\varrho, d^t_{h_k})$. For other slots in $(\tau_{a-1},\gamma]$ (where for $a = 0$ we let $\tau_{-1} = \delta$), the value of $\mathsf{pslack}(\tau)$ remains the same. (We will argue that invariant (InvP) is also preserved for slots $\tau \in (\delta,\tau_{a-1})$ when we process the group of segments that contains $\tau$.)

From the definitions of $f^*$, $g^*$, and invariant (InvP), we have that $\#\mathsf{pairs}(\tau) \le 0$ holds for $\tau \in (\delta, d_{f^*}) \cup [d_{g^*},\gamma]$, even after the step. Thus the claim holds in this range, which includes slots outside $(\delta,\gamma]$ if $d_{f^*} > \gamma$ or if $d_{g^*} \le \delta$ (note that either of the two conditions implies $d_{f^*} > d_{g^*}$). So for the rest of the proof we can assume that $\delta < d_{f^*} < d_{g^*} \le \gamma$ and that $\tau \in [d_{f^*}, d_{g^*})$.

Since the values of $\#\mathsf{pairs}(\tau)$ and $\mathsf{pslack}(\tau)$ change by at most 1, it is sufficient to show that each $\tau \in (\delta,\gamma]$ satisfies the following two conditions: (i) if $\#\mathsf{pairs}(\tau)$ increases then so does $\mathsf{pslack}(\tau)$, and (ii) if $\mathsf{pslack}(\tau)$ decreases then so does $\#\mathsf{pairs}(\tau)$.

To show (i), note that the case when $\#\mathsf{pairs}(\tau)$ increases happens when $d_g < d_f$ and $\tau \in [d_g, d_f)$. Since also $\tau \in [d_{f^*}, d_{g^*})$, this gives us that $g \ne g^*$ and $f \ne f^*$. Therefore $g = h_a$ and $f = \varrho$. If $a = k$, then $\mathsf{pslack}(\tau)$ also increases. Otherwise, $a < k$ and $d_{g^*} \le \tau_a$ by $g = h_a$ and by the definitions of $g^*$, $g$, and $a$. Since $\tau < d_{g^*}$, we get that $\tau \in [d^t_{h_a},\tau_a)$, which implies that $\mathsf{pslack}(\tau)$ increases as well.

To show (ii), suppose that $\mathsf{pslack}(\tau)$ decreases. This happens only when when $d_\varrho < d^t_{h_k}$ and $\tau \in [d_\varrho, d_{h_k})$. Since also $\tau \in [d_{f^*}, d_{g^*})$, we get $\tau \in [\max(d_\varrho, d_{f^*}), \min(d^t_{h_k}, d_{g^*}))$. Thus we only need to consider the case when $d_{g^*}$ is in segment $S'_k$ (that contains both $d_\varrho$ and $d^t_{h_k}$), which implies $a = k$ and $g = h_k$. As $f \in \{f^*,\varrho\}$, we have that $\tau \in [d_f, d_g)$, so $\#\mathsf{pairs}(\tau)$ decreases as well.

**Processing a middle group.** Let $[a,b]$ be a middle group; recall that $h_a \in \mathsf{ADV}$. We have two subcases.

<u>Case M.i</u>: There is $i \in [a,b]$ such that $w^t_{h_{i+1}} < \mu_i$, i.e., the algorithm increases the weight of $h_{i+1}$. Let $F(h_a)$ be the packet that is in a pair with $h_a$. We remove $F(h_a)$ from $\mathcal{F}$ and replace $h_a$ in $\mathsf{ADV}$ by a new shadow packet $s$ of weight $\mu_a = \mathsf{minwt}^t(\tau_a)$, added to the slot of $h_a$ in $\mathsf{ADV}$.

*Calculation showing* (5.15) *in Case (M.i).* We take into account the possible change of weights of $h_{a+1},\ldots,h_{b+1}$. By the case condition, there is $i \in [a,b]$ such that $w^t_{h_{i+1}} < \mu_i$, we thus use Lemma 5.4 with $a' = a+1$ and $b' = b+1$ to get $\Delta^t w(h_{a+1},\ldots,h_{b+1}) \le \mu_a - w^t_{h_{b+1}}$.

Next, observe that $w^t_{F(h_a)} \le w^t_{h_{b+1}}$. Indeed, $F(h_a)$ is not in $P$ and it was in a pair with $h_a$, thus $d^t_{F(h_a)} > \mathsf{prevts}^t(d^t_{h_a}) \ge \delta$ by Lemma 5.1(a). We get that $w^t_{F(h_a)} \le w^t_\varrho$ as $\varrho$ is the heaviest pending packet not in $P$ with deadline after $\delta$. Finally, $w^t_\varrho \le w^t_{h_{b+1}}$ implies $w^t_{F(h_a)} \le w^t_{h_{b+1}}$.

Then we prove (5.15) as follows:

$$\begin{aligned}
&\Delta_{[a,b]}\Psi - \phi\,\Delta^t w(h_{a+1},\ldots,h_{b+1}) - \mathsf{advgain}^t_{[a,b]}\\
&= \tfrac{1}{\phi}\left(\Delta^t w(h_{a+1},\ldots,h_{b+1}) - w^t_{F(h_a)} + w^t_{h_a}\right)\\
&\quad - \phi\,\Delta^t w(h_{a+1},\ldots,h_{b+1}) - (w^t_{h_a} - \mu_a)\\
&= -\tfrac{1}{\phi}\,w^t_{F(h_a)} - \tfrac{1}{\phi^2}\,w^t_{h_a} - \Delta^t w(h_{a+1},\ldots,h_{b+1}) + \mu_a\\
&\ge -\tfrac{1}{\phi}\,w^t_{h_{b+1}} - \tfrac{1}{\phi^2}\,w^t_{h_a} - (\mu_a - w^t_{h_{b+1}}) + \mu_a\\
&= -\tfrac{1}{\phi^2}\,w^t_{h_a} + \tfrac{1}{\phi^2}\,w^t_{h_{b+1}}\,,
\end{aligned}$$

where the last inequality follows from the aforementioned bounds.

*Invariant (InvP) in Case (M.i).* We claim that after we process the middle group, invariant (InvP) holds for slots in segments $S'_a, S'_{a+1}, \ldots, S'_b$ and it is not affected for another slot. Note that as $b < k$, the value of $\mathsf{pslack}()$ remains the same or increases for slots in segments $S'_a, S'_{a+1}, \ldots, S'_b$ (recall that changes of $\mathsf{pslack}()$ values in another segment are taken into account when we process the group containing that segment). We use Lemma 5.3 to analyze how the values of $\#\mathsf{pairs}$ change. If $d^t_{F(h_a)} \le d^t_{h_a}$, then $\#\mathsf{pairs}(\tau)$ decreases by 1 for $\tau \in [d^t_{F(h_a)}, d^t_{h_a})$. Otherwise, $d^t_{F(h_a)} > d^t_{h_a}$ and $\#\mathsf{pairs}(\tau) \le 0$ for $\tau \in [d^t_{h_a}, d^t_{F(h_a)})$. For other slots, $\#\mathsf{pairs}(\tau)$ remains the same. Hence, the claim holds.

<u>Case M.ii</u>: For all $i \in [a, b]$ we have $w^t_{h_{i+1}} \ge \mu_i$. Then the algorithm does not increase the weight of $h_{i+1}$ for any $i \in [a, b]$, i.e., $\Delta^t w(h_{a+1}, \ldots, h_{b+1}) = 0$. We replace $h_a$ in $\mathsf{ADV}$ by $h_{a+1}$, i.e., we put $h_{a+1}$ on the slot of $h_a$ in $\mathsf{ADV}$. Note that the new deadline of $h_{a+1}$ is $\tau_a$ and the new slot of $h_{a+1}$ in $\mathsf{ADV}$ is not after $\tau_a$.

We claim that $h_{a+1}$ is not in $\mathsf{ADV}$ before the replacement, therefore it is not twice in $\mathsf{ADV}$ after the replacement. This is trivial if $b > a$, since then packets $h_{a+1}, \ldots, h_b$ are not in $\mathsf{ADV}$ before processing the groups. Otherwise, we have $a = b$. Recall that we are processing groups from the last one to the first one, thus the group containing index $a + 1$ is already processed. Furthermore, we enforce that after processing a group $[a', b']$, packet $h_{a'}$ is not in $\mathsf{ADV}$, which shows the claim.

*Calculation showing* (5.15) *in Case M.ii.* We bound the cost of changes in the middle group $[a, b]$ by

$$\begin{aligned}
\Delta_{[a,b]}\Psi - \phi\Delta^t w(h_{a+1}, \ldots, h_{b+1}) - \mathsf{advgain}^t_{[a,b]} \\
&= \tfrac{1}{\phi}\left(w^t_{h_a} - w^t_{h_{a+1}}\right) - \phi \cdot 0 - (w^t_{h_a} - w^t_{h_{a+1}}) \\
&= -\tfrac{1}{\phi^2}\, w^t_{h_a} + \tfrac{1}{\phi^2}\, w^t_{h_{a+1}} \\
&\ge -\tfrac{1}{\phi^2}\, w^t_{h_a} + \tfrac{1}{\phi^2}\, w^t_{h_{b+1}}\,,
\end{aligned}$$

where that last inequality follows from $w^t_{h_{a+1}} \ge w^t_{h_{b+1}}$ as $a \le b$. This shows (5.15).

*Invariant (InvP) in Case M.ii.* We show that after we process the middle group, invariant (InvP) holds for slots in segments $S'_a, S'_{a+1}, \ldots, S'_b$ and it is not affected for another slot. Note that $\#\mathsf{pairs}(\tau)$ increases by 1 for $\tau \in [d^t_{h_a}, \tau_a)$ as $d^{t+1}_{h_{a+1}} = \tau_a$ and we replaced $h_a$ by $h_{a+1}$. The value of $\mathsf{pslack}(\tau)$ increases by 1 for $\tau \in [d^t_{h_a}, \tau_a)$, thus invariant (InvP) holds for such $\tau$. For other slots, the value of $\#\mathsf{pairs}()$ stays the same and the value of $\mathsf{pslack}()$ remains the same or increases.

**Processing the initial group.** If $i_1 > 0$ or if $\ell = 0$ and $\alpha > 0$, then there is the initial group $[0, b]$. Note that for any $i \in [0, b]$, $h_i \notin \mathsf{ADV}$. We do not change $\mathsf{ADV}$ or set $\mathcal{F}$, thus $\mathsf{advgain}^t_{[0,b]} = 0$ and $\#\mathsf{pairs}(\tau)$ remains the same for any slot $\tau$. Invariant (InvP) holds for a slot $\tau \in S'_0 \cup \cdots \cup S'_b$ as $b < k$ and as the value of $\mathsf{pslack}()$ does not decrease.

We need to estimate the change of the weights of packets $h_1, \ldots, h_{b+1}$, denoted $\Delta^t w(h_1, \ldots, h_{b+1})$. First, suppose that the algorithm increases the weight of at least one of the packets $h_1, \ldots, h_{b+1}$, i.e., there is $i \in [0, b]$ such that $w^t_{h_{i+1}} < \mu_i$. By Lemma 5.4 with $a' = 1$ and $b' = b + 1$ we have $\Delta^t w(h_1, \ldots, h_{b+1}) \le \mu_0 - w^t_{h_{b+1}}$.

Then the calculation showing (5.15) is simple:

$$\begin{aligned}
&\Delta_{[0,b]}\Psi - \phi\,\Delta^t w(h_1, \ldots, h_{b+1}) - \mathsf{advgain}^t_{[0,b]} \\
&= \tfrac{1}{\phi}\,\Delta^t w(h_1, \ldots, h_{b+1}) - \phi\,\Delta^t w(h_1, \ldots, h_{b+1}) - 0 \\
&= -\Delta^t w(h_1, \ldots, h_{b+1}) \\
&\ge -\mu_0 + w^t_{h_{b+1}} \\
&\ge -\tfrac{1}{\phi^2}\, w^t_p - \tfrac{1}{\phi}\, w^t_\varrho + w^t_{h_{b+1}} \\
&\ge -\tfrac{1}{\phi^2}\, w^t_p + \tfrac{1}{\phi^2}\, w^t_{h_{b+1}}\,,
\end{aligned}$$

where the penultimate inequality follows from $\mu_0 \le \frac{1}{\phi^2}\, w^t_p + \frac{1}{\phi}\, w^t_\varrho$ by (5.8) and the last one from $w^t_\varrho \le w^t_{h_{b+1}}$.

Otherwise, $\Delta^t w(h_1, \ldots, h_{b+1}) = 0$ and (5.15) holds, since its left-hand side is zero and the right-hand side is at most zero. This concludes the proof that the packet-scheduling inequality (5.4) holds in a leap step and also the proof of $\phi$-competitiveness of Algorithm $\mathsf{PlanM}$.

## 6 Final Comments

Our result establishes a tight bound of $\phi$ on the competitive ratio of $\mathsf{PacketScheduling}$ in the deterministic case, settling a long-standing open problem.

Among the remaining open problems in this area, the most prominent one is to establish tight bounds for randomized algorithms for $\mathsf{PacketScheduling}$. The best know upper bound to date is $\mathrm{e}/(\mathrm{e}-1) \approx 1.582$ [4, 9, 7, 17]. This ratio is achieved by a memoryless algorithm and it holds even against an adaptive adversary. No better upper bound for the oblivious adversary is known. (In fact, against the oblivious adversary the same ratio can be attained for a more general problem of online vertex-weighted bipartite matching [1, 13].) The best lower bounds are $4/3 \approx 1.333$ [7] against the adaptive adversary and 1.25 [10] against the oblivious one.

The determination of the packet to transmit needs to be made at speed matching the link's rate, so the running time and simplicity of the scheduling algorithm are important factors. This motivates the study of *memoryless* algorithms for $\mathsf{PacketScheduling}$, as those algorithms tend to be easy to implement and fast. All known upper bounds for competitive randomized algorithms we are aware of are achieved by memoryless algorithms (see [15]). For deterministic memoryless algorithms, the only one that beats ratio 2 is the 1.893-competitive algorithm in [14]. The main question here is whether the ratio of $\phi$ can be achieved by a memoryless algorithm.

**Acknowledgements** This work was supported by GA ČR project 17-09142S, GAUK project 634217, NSF grant CCF-1536026, and NCN grants 2016/21/D/ST6/02402 and 2016/22/E/ST6/00499. We are grateful to Martin Böhm for useful discussions.

## References

[1] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proc. of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '11)*, pages 1253–1264, 2011.

[2] William A. Aiello, Yishay Mansour, S. Rajagopolan, and Adi Rosén. Competitive queue policies for differentiated services. *J. Algorithms*, 55(2):113–141, 2005.

[3] Nir Andelman, Yishay Mansour, and An Zhu. Competitive queueing policies for QoS switches. In *Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '03)*, pages 761–770, 2003.

[4] Yair Bartal, Francis Y. L. Chin, Marek Chrobak, Stanley P. Y. Fung, Wojciech Jawor, Ron Lavi, Jiří Sgall, and Tomáš Tichý. Online competitive algorithms for maximizing weighted throughput of unit jobs. In *Proc. of the 21 st Symposium on Theoretical Aspects of Computer Science (STACS '04)*, volume 2996 of *LNCS*, pages 187–198. Springer Berlin Heidelberg, 2004.

[5] Marcin Bienkowski, Marek Chrobak, Christoph Dürr, Mathilde Hurand, Artur Jeż, Łukasz Jeż, and Grzegorz Stachowiak. Collecting weighted items from a dynamic queue. *Algorithmica*, 65(1):60–94, 2013.

[6] Marcin Bienkowski, Marek Chrobak, Christoph Dürr, Mathilde Hurand, Artur Jeż, Łukasz Jeż, and Grzegorz Stachowiak. A $\phi$-competitive algorithm for collecting items with increasing weights from a dynamic queue. *Theoretical Computer Science*, 475:92 – 102, 2013.

[7] Marcin Bienkowski, Marek Chrobak, and Łukasz Jeż. Randomized competitive algorithms for online buffer management in the adaptive adversary model. *Theoretical Computer Science*, 412(39):5121–5131, 2011.

[8] Martin Böhm, Marek Chrobak, Łukasz Jeż, Fei Li, Jiří Sgall, and Pavel Veselý. Online packet scheduling with bounded delay and lookahead. In *Proc. of the 27th International Symposium on Algorithms and Computation (ISAAC '16)*, volume 64 of *LIPIcs*, pages 21:1–21:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

[9] Francis Y. L. Chin, Marek Chrobak, Stanley P. Y. Fung, Wojciech Jawor, Jiří Sgall, and Tomáš Tichý. Online competitive algorithms for maximizing weighted throughput of unit jobs. *J. of Discrete Algorithms*, 4(2):255–276, 2006.

[10] Francis Y. L. Chin and Stanley P. Y. Fung. Online scheduling with partial job values: Does timesharing or randomization help? *Algorithmica*, 37(3):149–164, 2003.

[11] Marek Chrobak, Wojciech Jawor, Jiří Sgall, and Tomáš Tichý. Improved online algorithms for buffer management in QoS switches. In *Proc. of the 12th Annual European Symposium on Algorithms (ESA '04)*, volume 3221 of *LNCS*, pages 204–215, 2004.

[12] Marek Chrobak, Wojciech Jawor, Jiří Sgall, and Tomáš Tichý. Improved online algorithms for buffer management in QoS switches. *ACM Trans. Algorithms*, 3(4), 2007.

[13] Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. Randomized primal-dual analysis of RANKING for online bipartite matching. In *Proc. of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '13)*, pages 101–107, 2013.

[14] Matthias Englert and Matthias Westermann. Considering suppressed packets improves buffer management in quality of service switches. *SIAM Journal on Computing*, 41(5):1166–1192, 2012.

[15] Michael H. Goldwasser. A survey of buffer management policies for packet switches. *SIGACT News*, 41(1):100–128, 2010.

[16] Bruce Hajek. On the competitiveness of on-line scheduling of unit-length packets with hard deadlines in slotted time. In *Proc. of the 35th Conference on Information Sciences and Systems*, pages 434–438, 2001.

[17] Łukasz Jeż. A universal randomized packet scheduling algorithm. *Algorithmica*, 67(4):498–515, 2013.

[18] Łukasz Jeż, Fei Li, Jay Sethuraman, and Clifford Stein. Online scheduling of packets with agreeable deadlines. *ACM Trans. Algorithms*, 9(1):5:1–5:11, 2012.

[19] Alexander Kesselman, Zvi Lotker, Yishay Mansour, Boaz Patt-Shamir, Baruch Schieber, and Maxim Sviridenko. Buffer overflow management in QoS switches. *SIAM Journal on Computing*, 33(3):563–583, 2004.

[20] Fei Li, Jay Sethuraman, and Clifford Stein. An optimal online algorithm for packet scheduling with agreeable deadlines. In *Proc. of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '05)*, pages 801–802, 2005.

[21] Fei Li, Jay Sethuraman, and Clifford Stein. Better online buffer management. In *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '07)*, pages 199–208, 2007.

[22] An Zhu. Analysis of queueing policies in QoS switches. *J. Algorithms*, 53(2):137–168, 2004.