DIMACS Technical Report 98-10
February 1998

Patience is a Virtue: The Effect of Delay on Competitiveness for Admission Control

by

Michael H. Goldwasser[1]
Department of Computer Science
Princeton University
Princeton, NJ 08544
wass@cs.princeton.edu

[1]Permanent Member

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

We consider the problem of scheduling a single resource non-preemptively in order to maximize its utilization. The *delay* of a job is equal to the gap between its arrival time and the last possible time at which it may be started while still meeting its deadline. We introduce an additional restriction that each job must be willing to accept a delay proportional to its job length. That is, we assume there is some constant ω, such that any job of length $|J|$ allows a delay of at least $\omega \cdot |J|$. This restriction is quite natural for admission control, as it seems reasonable that a job requesting 5 minutes of a resource might be willing to wait at least 10 seconds before beginning if necessary, whereas a job requesting 5 hours of time should be willing to handle a wait of 10 minutes instead.

Our results show that this additional requirement has dramatic effects on the competitiveness of online algorithms. Without this restriction, previous lower bounds show that no algorithm, deterministic or randomized, can achieve any bounded competitiveness for the problem when arbitrarily job lengths are allowed. We show that for any $\omega > 0$ a simple greedy algorithm is $(2 + \frac{1}{\omega})$-competitive, and we give lower bounds showing that this is the best possible result for a deterministic algorithm, even when all jobs have one of three distinct lengths. In the special case where all jobs have the same length, previous results give a tight bound of 2 on the competitiveness for deterministic algorithms without a minimum delay. We generalize these results, showing that the competitiveness for any $\omega \geq 0$ is exactly $1 + \frac{1}{\lfloor \omega \rfloor + 1}$. We also give tight bounds for the case where jobs have one of two distinct lengths.

1 Introduction

We study a model for the online scheduling of a single resource. At arbitrary times, requests arrive for jobs that require the use of the resource for some specific amount of time and which must be completed by a given time or else be lost. The *delay* of that job is equal to the amount of time between the job's arrival and the last possible time at which it could be started in order to meet its deadline. For example, if a request comes for a job with delay zero, this job must either be given the resource immediately or else be lost for good. It is the role of the scheduler to decide which job to run on the resource through time, however we require that the schedule be *non-preemptive*, in that once a job has been started the scheduler is not allowed to stop the job. We do not necessarily require that the algorithm runs all jobs, but the goal of the scheduler is to maximize the amount of time for which the resource is in use. Our paper introduces an additional requirement, namely that every job of length $|J|$ must be willing to handle a delay of at least $\omega|J|$, for some constant $\omega \geq 0$.

The problem is online in nature, as we assume that the scheduler has no knowledge about the existence of any job until the time at which a request arrives. However when a request does arrive, the algorithm is immediately told both the length and deadline of the job. We will analyze the performance of an algorithm using *competitive analysis*, comparing the schedule produced by the algorithm to an optimal schedule produced by an algorithm that has full knowledge of all jobs from the beginning [7, 11].

This framework can be used to model many applications, and typically is motivated mostly by applications of scheduling for the delivery of communication through a network. For example, a video-on-demand provider may have a channel that can be devoted to sending video to a user, and requests may arrive requiring the channel for a specific length of time, to begin before a given deadline. We feel that the requirement of minimum delays proportional to the job length is quite reasonable in many settings, as someone requesting the resource for a long period of time should be more willing to wait longer than someone who may simply want one minute's use of the resource. A special case of this problem, studied by Lipton and Tomkins, is when all jobs have delay exactly zero, and thus immediately need to be scheduled or else lost [8]. This model was then generalized to include delays by Goldman *et. al.*,, where they allow each job to specify an arbitrary delay [6]. Notice that the existence of delays, in general, acts as a double-edged sword. Clearly, the existence of delays can only help a scheduler in increasing the resource utilization, as it offers more flexibility for scheduling jobs. However in terms of competitive analysis, delays may create more difficulties for an algorithm, as the optimal offline algorithm may know how to take better advantage of the flexibility. A requirement that minimum delays be proportional to the job lengths was introduced into a different model for scheduling on networks, where lower bounds are provided showing that this requirement does not significantly improve the overall competitiveness of the problem [5].

The main result of our paper is that for this problem, requiring a job's delay to be at least proportional to the job's length significantly improves the competitiveness for online scheduling algorithms. When $\omega = 0$, a previous lower bound shows that the competitiveness

depends logarithmically on the maximum job length [8], even when randomization is allowed, and thus no bounded competitiveness is possible when arbitrary job lengths are allowed. We show that a simple deterministic algorithm is $(2 + \frac{1}{\omega})$-competitive for all values of $\omega > 0$, thereby providing a constant competitive ratio which does not depend on the maximum length job. We provide a lower bound showing that this is the best possible deterministic result, even when all jobs have one of three distinct lengths. In the special case where all jobs are required to have the same length (e.g., packets in an ATM network) [6], we generalize previous results, showing that the natural greedy algorithm is $1 + \frac{1}{\lfloor\omega\rfloor+1}$-competitive, and again that this is the best possible deterministic result. In the case where jobs have one of two distinct lengths, we give tight bounds improving slightly on the $(2 + \frac{1}{\omega})$ bound for arbitrary lengths.

Our presentation is as follows. In Section 2, we begin by reviewing, in more detail, related research from the scheduling community. We define the precise model and notation in Section 3, and we give a summary of our exact results in Section 4. We present our upper bounds in Section 5, where we prove the competitiveness for several simple greedy algorithms. Following this, in Section 6, we provide lower bounds showing that, with one exception, our results are tight for all values of ω. We discuss open directions for research in Section 7. Finally, in the appendix, we provide more detailed results for the competitiveness when parameterized by both the value of ω, as well as the ratio between minimum and maximum job lengths.

2 Previous Work

The problem of scheduling a single resource non-preemptively, when all jobs have delay zero was studied by Lipton and Tomkins [8]. In the case where all jobs have one of two lengths 1 or Δ, they provide a randomized online algorithm that is 2-competitive for any Δ, and they give a lower bound that this is the best possible result. When arbitrary job lengths are considered, they provide a lower bound of $\Omega(\log \Delta)$ for the competitiveness of any randomized algorithm, where Δ is the ratio between the longest and shortest length. In this sense, they show that it is not possible to have an algorithm that has bounded competitiveness when Δ is unrestricted. They provide an $O(\log^{1+\epsilon} \Delta)$-competitive randomized algorithm for this case. This work generalized an earlier deterministic lower bound by Long and Thakur for a related scheduling model in the context of scheduling disk transfers [9].

Goldman, Parwatikar and Suri extended the model of Lipton and Tomkins to allow jobs to specify arbitrary delay times, up to which they are willing to wait [6]. When arbitrary delay times are allowed, their model is exactly the same as the case $\omega = 0$ for our model. For arbitrary job lengths, they give a randomized algorithm that improves upon the result of Lipton and Tomkins, providing a $6(\lceil \lg_2 \Delta \rceil + 1)$-competitive algorithm. When two job lengths are allowed, they provide a 4-competitive randomized algorithm. Additionally, they consider the case where all job lengths are identical (a case that is easily solvable without any delays). They show that a deterministic greedy algorithm is 2-competitive, and that this is the best possible result. They give a lower bound of 4/3 for the randomized competitiveness

in this setting, however without a matching upper bound. Additionally, they prove a curious result, namely that if all delays are equal to 1 or greater, then the same greedy algorithm becomes 3/2-competitive. It is this result which is really the springboard of our own work. Our work generalizes this result for all values of ω, and then we go on to consider the cases where job lengths vary. Goldman *et. al.*, consider one other model for delays that they name *uniform delays*. In this setting, they require that all jobs of the same length have the same delay, although the value of this delay need not have anything to do with the job length. With this additional requirement, they are able to improve upon some of their results because there is no longer an issue of deciding how to choose between two jobs of the same length. This model of uniform delays does not readily translate to our own model, however our results are significantly stronger than the results for uniform delays. Baruah *et. al.* give a constant competitive algorithm for maximizing the resource utilization in a similar model as ours, except where they allow *preemption* of calls [3].

Our single resource scheduling problem is simply a special case of the more general problem of admission control of calls in larger communication networks. A general model for virtual circuit routing consists of a communication resource for each edge in a network, and requests arrive asking for a connection between two different points in the network, while specifying parameters such as the duration of the call, the bandwidth required, and possibly other information depending on the model. Awerbuch, Azar and Plotkin provide a deterministic, non-preemptive $O(\log \Delta)$-competitive algorithm for maximizing the throughput on a network, where they require an additional bound on the bandwidth used by a single call [1]. They also provide a lower bound showing that this is the best possible result for a deterministic algorithm. The restriction on bandwidth was removed for the special case of tree networks by Awerbuch, Bartal Fiat and Rosén [2].

Finally, an interesting result in relation to our work, is that of Feldman *et. al.* [5]. They consider the requirement of minimum delays proportional to job lengths, in the more general context of non-preemptive call control on networks. They generalize previous $\Omega(\log n)$ lower bounds, showing that even when a network is a linear array and randomization is allowed, the minimum delay requirement can not be used to asymptotically improve the competitiveness of algorithms.

3 Notation

We consider a job (or call) J_i to be a triple of positive numbers $\langle a_i, l_i, w_i \rangle$, where a_i is the arrival time of the job, l_i is the processing time, and w_i is the amount of time the job is willing to wait after its arrival, before it must begin[6]. We will also refer to the *deadline* of the job, $d_i = a_i + w_i + l_i$, as the time at which the job must be completely processed, and the *expiration time*, $x_i = a_i + w_i$, as the latest time at which a job can be started in order to meet its deadline. We say that a job J_i is *available* at time t with respect to a partial schedule σ, if $a_i \leq t \leq x_i$, and if job J_i has not been started in σ prior to time t. The *gain* of a schedule σ is equal to $|\sigma| = \sum_{J \in \sigma} |J|$, and the goal is to maximize the gain.

Without loss of generality, we assume that all job lengths are real numbers lying in the

range $[1..\Delta]$, for some $\Delta \geq 1$. Furthermore, for constant $\omega \geq 0$, we require that every job J_i with length l_i has a delay $w_i \geq \omega l_i$. Our results are robust, in the sense that our lower bounds apply even when both ω and Δ are given as part of the input, whereas our upper bounds are achieved using algorithms that need not know either of ω or Δ in advance.

3.1 Competitive Analysis

We use competitive analysis to measure the quality of an online algorithm [7, 11]. The performance of an online algorithm is evaluated by comparing the gain of the algorithm to the gain of the optimal offline algorithm that knows the entire future when making decisions. For (deterministic) online algorithm A, and problem instance $\mathcal{I}$ we call $\mathcal{C}_A(\mathcal{I}) = \text{gain}_{opt}(\mathcal{I})/\text{gain}_A(\mathcal{I})$ the competitiveness of algorithm A on instance $\mathcal{I}$. The overall competitiveness of algorithm A is $\mathcal{C}_A = \max_{\mathcal{I}} \mathcal{C}_A(\mathcal{I})$, and so we call an algorithm *c-competitive*[1] if for all instances $\mathcal{I}$, $\text{gain}_{opt}(\mathcal{I}) \leq c \cdot \text{gain}_A(\mathcal{I})$. Finally, we define the competitiveness of the problem, $\mathcal{C} = \min_A \mathcal{C}_A$, to be the competitiveness of the best possible (deterministic) online algorithm, and we say an algorithm is *strongly competitive* if it is $\mathcal{C}$-competitive. If no algorithm exists with bounded competitiveness, we say $\mathcal{C} = \infty$. We can also consider the use of randomized online algorithms. In this case, the competitiveness compares the gain of the optimal schedule to the *expected* gain of the randomized algorithm. We assume that the input for an algorithm is chosen by an *oblivious* adversary, who must choose the entire sequence before the algorithm begins its work [4, 10].

In order to characterize the competitiveness of this scheduling problem based on different values of Δ and ω, we introduce the following notation. We let $D(\omega, \Delta)$ be the deterministic competitiveness of the scheduling problem for a fixed $\omega \geq 0$ and $\Delta \geq 1$. By definition, we see that $D(\omega, \Delta') \leq D(\omega, \Delta)$ for $\Delta' < \Delta$, and that $D(\omega', \Delta) \leq D(\omega, \Delta)$ for $\omega' > \omega$ as in each case, the left term refers to a more restricted set of inputs then the right term. The value $D(0, \Delta)$ is thus the competitiveness of the problem for a fixed Δ without any restriction on the delays. We use the notation $\hat{D}(\omega) = \max_\Delta D(\omega, \Delta)$ to denote the competitiveness of the problem for a fixed ω without any restriction on the maximum length. Finally, $\hat{D}(0)$ is thus the competitiveness of the problem without either restriction.

As was done by previous researchers, we study the special cases where all jobs have unit length, or where all jobs are of one of two lengths. We define $D_1(\omega)$ to be the deterministic competitiveness for the case where all jobs must have length 1, and we define $D_2(\omega, \Delta)$ to be the competitiveness for the case where all jobs must have either length 1 or length Δ. In general, we see that $D(\omega, \Delta) \geq D_2(\omega, \Delta) \geq D_1(\omega)$ as each expression from left to right refers to a more restrictive set of problem instances. We again[2] define $\hat{D}_2(\omega) = \max_\Delta D_2(\omega, \Delta)$.

[1] Classically, the notion of competitiveness allows for a small additive error, namely that $\text{gain}_{opt}(I) \leq c \cdot \text{gain}_A(\mathcal{I}) + b$ for some constant b. However, for this scheduling problem the definitions are identical because we can magnify any additive error by creating multiple copies of the same input placed end to end so as not to interfere with each other.

[2] As a technicality, notice that we cannot claim that$D_2(\omega, \Delta') \leq D_2(\omega, \Delta)$ for $\Delta' < \Delta$, as we have defined $D_2(\omega, \Delta)$ such that all jobs *must* be of length 1 or Δ. Therefore a construction that uses lengths 1 and Δ' is not a valid instance of the problem $D_2(\omega, \Delta)$.

Finally, we characterize the competitiveness of randomized algorithms versus an oblivious adversary. We define $R(\omega, \Delta)$, $R_2(\omega, \Delta)$ and $R_1(\omega)$ to be the randomized competitiveness for the corresponding situations. Since any deterministic algorithm can be viewed as a randomized algorithm, we immediately have that $R(\omega, \Delta) \leq D(\omega, \Delta)$ for all cases.

3.2 A recap of previous work

We review the results of previous researchers when translated to our notation. Lipton and Tomkins prove the lower bound, $\hat{R}(0) \geq 2$, by giving a proof that implies that for any fixed Δ, $R_2(0, \Delta) \geq 2 - \frac{1}{\Delta}$ [8]. When arbitrary length jobs are allowed, they prove a lower bound of $R(0, \Delta) = \Omega(\log(\Delta))$, and thus $\hat{R}(0) = \infty$. For unit-length jobs, Goldman, Parwatikar and Suri prove a tight bound of $D_1(0) = 2$, and they also show that $D_1(1) \leq \frac{3}{2}$ and that $R_1(0) \geq 4/3$ [6]. When jobs may have two distinct lengths, they show $\hat{R}_2(0) \leq 4$, and when arbitrary lengths are allowed, they show $R(0, \Delta) \leq 6(\lceil \lg_2 \Delta \rceil + 1)$, matching the Lipton-Tomkins lower bound up to a constant factor.

4 Summary of results

When all jobs are of unit length, we prove that the natural Greedy algorithm is a strongly $(1 + \frac{1}{\lfloor \omega \rfloor + 1})$-competitive deterministic algorithm for all values of ω. This generalizes the previous results for the case $\omega = 0$. For the case of two distinct job lengths, we give tight bounds showing that for all values of $\omega > 0$, the deterministic competitiveness is exactly $1 + \max\left(\frac{\lceil \omega \rceil + 1}{\lceil \omega \rceil}, \frac{\lfloor \omega \rfloor + 1}{\omega}\right)$. When arbitrary job lengths are allowed, we show that a greedy algorithm is $(2 + \frac{1}{\omega})$-competitive for all $\omega > 0$. We give a lower bound showing that this value is tight for all values of ω, even if only three distinct job lengths are allowed.

A complete summary of our upper and lower bounds when parameterized by ω is given in Table 4. A more complete set of bounds, when parameterized by both ω and Δ, is given in the appendix.

5 Deterministic Online Algorithms

We define a *greedy-type* algorithm to be one that never sits idle while a job is available. For such an algorithm, there is still some decision as to which job it chooses to run when several are available. When all jobs have unit length, without loss of generality, it is clear that the best job to run is the available job with the earliest expiration time, and so we call this the Greedy algorithm. When there exist jobs of two lengths, we will analyze another natural greedy algorithm. When there are jobs of arbitrary lengths, there are several natural choices for which job to run. Although some rules may be better than others, we prove in Section 5.3 that any *greedy-type* algorithm is strongly competitive for all values of ω.

For these proofs, we borrow the notions of *blocking* and *covering* from [6]. Given two schedules σ and σ^*, assume that job $J_j \in \sigma$ begins running in σ at time J_j^σ, and that

Unit-length jobs				
	$D_1(\omega)$		$R_1(\omega)$	
	LB	UB	LB	UB
$\omega = 0$	2 From [6]	From [6]	$4/3$ From [6]	
$\omega \geq 0$	$1 + \frac{1}{\lfloor\omega\rfloor+1}$ (Thm. 12)	(Thm. 4)	$1 + \frac{1}{2\lfloor\omega\rfloor+3}$ (Thm. 12)	

Two distinct job lengths				
	$\hat{D}_2(\omega)$		$\hat{R}_2(\omega)$	
	LB	UB	LB	UB
$\omega = 0$	∞ (Cor. 16)		2 From [8]	4 From [6]
$\omega > 0$	$1 + \max(\frac{\lceil\omega\rceil+1}{\lceil\omega\rceil}, \frac{\lfloor\omega\rfloor+1}{\omega})$ (Cor. 17)	(Cor. 9)		

Arbitrary job lengths				
	$\hat{D}(\omega)$		$\hat{R}(\omega)$	
	LB	UB	LB	UB
$\omega = 0$	∞ (Cor. 21)		∞ From [8]	
$\omega > 0$	$2 + \frac{1}{\omega}$ (Cor. 22)	(The. 10)		

Table 1: The lower/upper bounds for the deterministic/randomized competitiveness of the problem, based on specific values of ω. Randomized upper bounds are only shown when they improve on the deterministic results.

job $J_i \in \sigma^*$ begins running in σ^* at time $J_i^{\sigma^*}$, where i and j may or may not differ. We say that job J_j^σ *blocks* $J_i^{\sigma^*}$ if J_j is running in σ at the time when J_i begins in σ^*, namely $J_j^\sigma \leq J_i^{\sigma^*} < J_j^\sigma + |J_j|$. If job J_j^σ blocks $J_i^{\sigma^*}$, we additionally say that J_j^σ *covers* $J_i^{\sigma^*}$ if J_j is still running in σ after J_j ends in σ^*, namely $J_j^\sigma + |J_j| \geq J_i^{\sigma^*} + |J_i^{\sigma^*}|$.

As part of our proofs, we will often need to prove an upper bound on the ratio $\frac{|\mathcal{J}'|}{|\mathcal{J}|}$ between the total length of jobs in a set $\mathcal{J}'$ to the total length of jobs in a set $\mathcal{J}$ (for example to compare the jobs scheduled in an optimal schedule versus the jobs scheduled by some algorithm). To do so, we use a charging scheme, similar to those of [6, 8]. In general, we will devise charging schemes where every job $J \in \mathcal{J}$ distributes up to $c|J|$ units of value to various jobs of $\mathcal{J}'$. If we can then show that every job $J' \in \mathcal{J}'$ has received at least $c'|J'|$ units of cost, then we have proven that $c|\mathcal{J}| \geq c'|\mathcal{J}'|$, that is $\frac{|\mathcal{J}'|}{|\mathcal{J}|} \leq \frac{c}{c'}$.

5.1 Unit-Length Jobs

We consider the deterministic GREEDY algorithm that always schedules the available job with the earliest expiration time. We show that GREEDY achieves a competitive ratio of $1+\frac{1}{\lfloor\omega\rfloor+1}$. Later, we will see that this is in fact a strongly-competitive deterministic algorithm, for all values of ω.

As in [6], we consider the schedule σ produced by GREEDY, and call the periods during which the resource is continuously in use the *busy periods* of σ. Label these periods as $\pi_1, \pi_2, \ldots, \pi_m$, and let b_i and e_i, respectively, denote the times at which π_i begins and ends. Partition the job sequence $\mathcal{S}$ into classes $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$, where $\mathcal{S}_i$ consists of exactly those jobs that arrived during the period $[b_i, e_i)$. In order to prove that GREEDY is c-competitive, it is sufficient to prove this result over subsequence $\mathcal{S}_1$. Without loss of generality, we can artificially push back the arrival times of all other jobs to be later than the latest deadline of a job in $\mathcal{S}_1$. This change will have no effect on the gain for GREEDY and the additional time can only help the optimal offline algorithm.

So without loss of generality, from this point on we assume that our instances are such that the GREEDY schedule results in a single busy period. We denote as σ, the schedule produced by GREEDY, and we denote as σ^*, an arbitrary optimal schedule. We also let σ and σ^* denote the actual set of jobs successfully run in the respective schedule. If we assume that the first job arrives at time $t = 0$, then each job in σ will be started at an integral time unit.

We let $\mathcal{L}$ denote the jobs in σ^* scheduled to begin on or after time $e = e_1$ (i.e. 'late'), $\mathcal{P}$ denote the jobs in σ^* scheduled to begin strictly before time e (i.e. 'prompt'), and $\mathcal{N}$ denote the jobs in σ that are not members of $\mathcal{L}$. We denote $|\mathcal{L}| = L$, $|\mathcal{P}| = P$, and $|\mathcal{N}| = N$. By definition, we see that $|\sigma^*| = P + L$. We claim that $\mathcal{L} \subseteq \sigma$, as every job of $\mathcal{L}$ was started in σ^* at time e or later, and hence has expiration time at least e. GREEDY's queue was empty from time e on, so it must be the case that any such job was already run in σ. Therefore σ is exactly $\mathcal{N} \cup \mathcal{L}$ and so $|\sigma| = N + L$.

Rather than analyze the true schedule σ, we construct a different schedule σ', and base our analysis on it. Given both σ and σ^*, we construct σ' as follows. We filter through the

GREEDY schedule σ from beginning to end, and at any starting point t where σ starts a job J that was never run in σ^*, we look at the set $\mathcal{A}$ of all jobs from σ^* that are currently available. If $\mathcal{A}$ is non-empty, we throw away J for good, and replace the spot in the schedule with the member J_a of $\mathcal{A}$ with the earliest expiration. If J_a was originally scheduled in σ at a later time, then we leave the abandoned spot empty for now and allow it to filled later, if possible, as we filter that part of the original schedule. We call σ' the resulting schedule at the end of this process. Notice that by construction, $|\sigma'| \leq |\sigma|$. All elements of $\mathcal{L}$ are still in σ', as our construction would never have thrown out any elements of $\mathcal{L} \subseteq \sigma^*$. If we analogously define $\mathcal{N}'$ as $\sigma' - \mathcal{L}$, it must be that $N' \leq N$. We rely on the following two technical lemmas, whose proofs are delayed until Section 5.1.1.

Lemma 1 *For any time t at which σ originally started a job, if $J_k \in \sigma^* - \sigma'$ is available to σ', then σ' must start some job $J_i \in \sigma^*$ at time t, with $x_i \leq x_k$.*

Lemma 2 *If σ' starts some job at a given time, there can not exists two jobs of $\sigma^* - \sigma'$, both of which are available with respect to σ' at that time.*

Now, we begin our analysis of GREEDY. Recall that $|\sigma^*| = P + L$ and $|\sigma| = N + L$. Furthermore, it must be that $P \leq N + L$, as GREEDY runs $N + L$ jobs continuously from time 0 to time e, and thus the number of jobs that the optimal schedule begins strictly before time e can be at most this many.

Lemma 3

$$N \geq \frac{\lfloor \omega \rfloor}{\lfloor \omega \rfloor + 1} P.$$

Proof: Recalling that $N \geq N'$, we devise a charging scheme where each job of $\mathcal{N}'$ is allowed to distribute up to $\lfloor \omega \rfloor + 1$ units of charge, and each job of $\mathcal{P}$ receives at least $\lfloor \omega \rfloor$ units.

Given job $J_i \in \mathcal{N}'$ that runs in σ' at time t, we assign costs as follows:

- $\lfloor \omega \rfloor$ is assigned to itself, if this job also run in σ^*.
- 1 is assigned to any job $J_k \in \sigma^* - \sigma'$, that was available to σ' at time t.

It is clear that each job of $\mathcal{N}'$ has assigned at most $\lfloor \omega \rfloor + 1$ units overall. This is a result of Lemma 2, that assures us that at most one job will be paid by the second assignment rule.

Now, we show that all jobs of $\mathcal{P}$ have been assigned at least $\lfloor \omega \rfloor$. If a job $J_k \in \mathcal{P}$ was run in σ', then that job will have been assigned $\lfloor \omega \rfloor$ from itself, and we are done. We must only show this result for a job $J_k \in \mathcal{P}$ that did not ever run in σ'. First we note that such a job must not be scheduled in σ either (our construction would never have thrown out a member of $\mathcal{P}$), and thus it must be that $x_k < e$.

Since this job has a minimum delay of at least ω, there must have been at least $\lfloor \omega \rfloor$ original start times at which this job was available to σ'. By Lemma 1, σ' must start some

job J_i with $x_i \leq x_k < e$, and thus J_i is not a late job, rather $J_i \in \mathcal{N}'$. By the second assignment rule, it must be that J_i paid 1 to J_k. Since this is true at each of the at least $\lfloor \omega \rfloor$ time slots at which J_k was available to σ', it must be that J_k has been assigned at least $\lfloor \omega \rfloor$. ∎

Theorem 4 *The* GREEDY *algorithm is* $\left(1 + \frac{1}{\lfloor \omega \rfloor + 1}\right)$*-competitive, thus* $D_1(\omega) \leq \left(1 + \frac{1}{\lfloor \omega \rfloor + 1}\right)$.

Proof: This follows from Lemma 3.

$$\begin{aligned} N &\geq \frac{\lfloor \omega \rfloor}{\lfloor \omega \rfloor + 1} P = P - \frac{1}{\lfloor \omega \rfloor + 1} P \geq P - \frac{1}{\lfloor \omega \rfloor + 1}(N + L) \\ P &\leq N + \frac{1}{\lfloor \omega \rfloor + 1}(N + L) \\ P + L &\leq N + L + \frac{1}{\lfloor \omega \rfloor + 1}(N + L) = \left(1 + \frac{1}{\lfloor \omega \rfloor + 1}\right)(N + L) \\ \frac{|\sigma^*|}{|\sigma|} = \frac{P + L}{N + L} &\leq \left(1 + \frac{1}{\lfloor \omega \rfloor + 1}\right) \end{aligned}$$

∎

5.1.1 Technical lemmas

Lemma 1 *For any time t at which σ originally started a job, if $J_k \in \sigma^* - \sigma'$ is available to σ', then σ' must start some job $J_i \in \sigma^*$ at time t, with $x_i \leq x_k$.*

Proof: Let J_k be a job run in σ^*, but not σ', and let t be some original start time of σ at which J_k was available to σ'. If this time slot were empty or contained a job not in σ^*, our construction surely would have placed J_k or some other job of σ^* in this slot instead. So some job $J_i \in \sigma^*$ must be started in σ'.

J_i's placement in σ' could be for one of two reasons. If J_i was originally scheduled in this slot from σ, it must be that $x_i \leq x_k$ since GREEDY could have chosen J_k. If J_i was inserted during the transformation to σ', then it also must be that $x_i \leq x_k$, since our construction chose the available item of σ^* with minimal expiration, and so certainly J_k was a candidate. Therefore, we have that $x_i \leq x_k$. ∎

Lemma 2 *If σ' starts some job at a given time, there can not exists two jobs of $\sigma^* - \sigma'$, both of which are available with respect to σ' at that time.*

Proof: Suppose J_1 and J_2 are two such jobs from $\sigma^* - \sigma'$, both of which are available to σ' at the same point in time. Let us assume that a is the earlier arrival time of the two jobs,

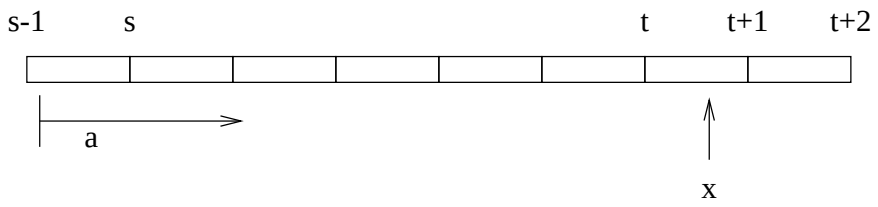

Figure 1: The range $[s, t]$ in σ'.

and that x is the later expiration of the two jobs. Let t be the latest time at which σ' started a job, even though either J_1 or J_2 was available (and thus $t \leq x < t + 1$). By Lemma 1, the job scheduled at time t must have had an expiration time less than or equal to x and must also be from σ^*.

Now, let s be the earliest start time in σ', such that over the range $[s, t]$, all jobs that σ' starts have an expiration time of x or earlier, and also are jobs from σ^*. Since the intervals of availability for our two jobs overlap, then at least one of them was available to σ' throughout the range $[a, t]$. Therefore, Lemma 1 assures that every job started in this range is a member of σ^* and has an expiration time of x or less. This implies that $s \leq \lceil a \rceil \leq a + 1$. A sample range $[s, t]$ is displayed in Figure 1.

Let us denote as R, the set of all jobs that σ' began during $[s, t]$, together with jobs J_1 and J_2. Notice that $|R| = 2 + (t - s) + 1 = t - s + 3$. We have already seen that every job of R has an expiration $x_i \leq x < t + 1$, and belongs to σ^*. We also claim that each of these jobs has an arrival time $a_i > s - 1$. By our choice of s, any job that starts at time $s - 1$ in σ', either has an expiration strictly greater than x or else is not a member of σ^*. However, if a job in R had arrived on or before $s - 1$, the job currently starting at $s - 1$ would not be there. If the current job is a member of $\sigma' - \sigma^*$, then it certainly would have been replaced during the transformation of σ to σ', and if the current job has expiration greater than x, then it would not have been chosen as the job with minimum expiration during either the original GREEDY algorithm or during the transformations.

At this point, we have a set of $t - s + 3$ jobs, each of which arrives strictly after time $s - 1$ and must start strictly before time $t + 1$. However for any schedule, at most $s - t + 2$ jobs can possibly start during the interval $(s - 1, t + 1)$, and thus it cannot be the case that the optimal schedule runs all of R. This is a contradiction, as all members of R are contained in σ^*. ∎

5.2 Jobs of Two Distinct Lengths

We consider the following deterministic algorithm, which we call GREEDY-TWOLENGTHS. When the resource is free, if there exists an available large job, the algorithm schedules the available large job with the earliest expiration time. If no large jobs are available, then the algorithm schedules the available small job with the earliest expiration time.

Theorem 5 *For all ω and Δ, if an algorithm never allows the resource to remain idle while jobs are available, that algorithm is* $\max(1+\Delta, 2+\frac{1}{\Delta})$*-competitive.*

Proof: We give the following charging scheme where each job $J \in \sigma$ distributes at most $c|J|$ units, and each job $J' \in \sigma^*$ receives at least $|J'|$ units of value, where $c = \max(1+\Delta, 2+\frac{1}{\Delta})$.

Every job $J \in \sigma$, pays $|J|$ to itself if it also ran in σ^*, and pays $|J'|$ to any job $J' \in \sigma^*$ that is either blocked or covered by J. From these assignments, it is immediate that any job $J' \in \sigma^*$ was paid at least $|J'|$. If job J' ran at any point in σ, then it will have paid for itself. Otherwise, consider the time t at which J' began in σ^*. Since J' never ran in σ, it was available at time t. Because our algorithm never leaves the resource idle if a job is available, then some other job must either start or be in progress at time t in σ. But any such job either blocks or covers J', and so we again see that J' will have been paid.

The more difficult fact to show is that any job $J \in \sigma$ pays out at most $c|J|$ units. We analyze the small and large jobs separately. For a small job J, we note that it cannot possibly cover another job, and it can only block one other job. Therefore, the most it could have possibly paid is $1+\Delta$. But since $1+\Delta \leq c$, we see that it has paid at most $c = c|J|$ units. For a large job J, it may cover up to $\lfloor\Delta\rfloor$ small jobs, as well as paying for itself and at most one additional job that it may block. Therefore, the most that it can pay out is $\Delta + \Delta + \lfloor\Delta\rfloor = |J|\left(2 + \frac{\lfloor\Delta\rfloor}{\Delta}\right)$. Note that if $\Delta \geq 2$, then we see that $c \geq \Delta + 1 \geq 3$, and therefore, $2 + \frac{\lfloor\Delta\rfloor}{\Delta} \leq c$. If $1 < \Delta < 2$, we see that $2 + \frac{\lfloor\Delta\rfloor}{\Delta} = 2 + \frac{1}{\Delta} \leq c$. Therefore J has paid at most $c|J|$. ∎

Theorem 6 *For $\omega \geq \frac{1}{\Delta}$, the* Greedy-TwoLengths *algorithm is* $\left(2 + \frac{\lceil\Delta\rceil - 1}{\Delta}\right)$*-competitive.*

Proof: The key insight to this proof is that since $\omega \geq \frac{1}{\Delta}$, then the minimum delay for a large job $\omega\Delta \geq 1$. Therefore, it is not possible for a single small job to completely block the available window for a large job. If a small job is in progress when some large job arrives, that large job will still be available when the small job ends. At that time, if that large job is not the next to run, it will certainly be the case that the job which does run is also large. If our large job expires without being scheduled in this case, it will be able to blame both the small job and this other large job. With this in mind, we say that a large job $J_j \in \sigma$ *pseudo-blocks* a large job $J_i \in \sigma^*$, if J_j begins in σ strictly within the first unit of time of when J_i runs in σ^*, or more formally if $J_i^{\sigma^*} < J_j^{\sigma} < J_i^{\sigma} + 1$.

Now we prove the claim, describing a charging scheme where each job $J \in \sigma$ distributes at most $\left(2 + \frac{\lceil\Delta\rceil - 1}{\Delta}\right)|J|$ units, and each job $J' \in \sigma^*$ receives at least $|J'|$ units of value.

Every small job $J \in \sigma$, pays 1 to itself, and 1 to any job $J' \in \sigma^*$, large or small, that is blocked by J. Every large job $J \in \sigma$, pays the following:

- Δ to itself
- Δ to any large job $J' \in \sigma^*$, that is blocked by J
- 1 to any small job $J' \in \sigma^*$, that is blocked by J

- $\Delta - 1$ to any large job $J' \in \sigma^*$, that is pseudo-blocked by J

We first claim that every job $J' \in \sigma^*$ collects at least $|J'|$ units. Any such job which also runs in σ will have paid for itself. If such a small job does not run in σ, then there must be some other job which blocks it, and that job will have paid 1 to J'. If a large job J' does not run in σ, we know that it must be blocked by some other job. If it was blocked by a large job, then that job will have paid $\Delta = |J'|$ to J'. Otherwise, J' was blocked by a small job, which will have paid 1 to J'. However in this case, we claim that there must be some large job which *pseudo-blocks* J', and thus pays $\Delta - 1$ to J', in which case J' has received $\Delta = |J'|$ overall. Let J_s be the small job which blocks J'. Since our algorithm will never run a small job when a large is available, if J_s^σ denotes the time at which J_s starts in σ, it must be the case that J' arrives in the range $(J_s^\sigma, J_s^\sigma + 1)$. Since we have seen that the delay of a large job is at least 1, then at time $J_s^\sigma + 1$ job J' must still be available. Therefore, the job started then by the GREEDY-TWOLENGTHS algorithm must be some other large job, and it is exactly the case that this large job pseudo-blocks J'.

Now we claim that every job $J \in \sigma$ pays out at most $\left(2 + \frac{\lceil\Delta\rceil - 1}{\Delta}\right)|J|$ units. When J is a small job, in fact, it pays out at most 2 units, as it can block at most one job, and thus pay 1 each to the blocked job and to itself. The more interesting case is when J is a large job, and thus we must show that it pays at most $\left(2 + \frac{\lceil\Delta\rceil - 1}{\Delta}\right)\Delta = 2\Delta + \lceil\Delta\rceil - 1$ units. Assume that J runs in σ during the interval $[t, t + \Delta)$, and we consider two cases, depending on whether J pseudo-blocks some other job or not. If J does not pseudo-block any job, then at most $\lceil\Delta\rceil$ jobs can begin during $[t, t + \Delta)$ in σ^* and thus be blocked, and at most one of these blocked job can be large. Therefore, J pays at most Δ to itself, and $\Delta + \lceil\Delta\rceil - 1$ to blocked jobs. In the case where J does pseudo-block some other large job, we notice that the pseudo-blocked job must be running throughout the closed interval $[t, t + \Delta - 1]$ of σ^*, and therefore at most one other job can begin in the interval $(t + \Delta - 1, t + \Delta)$, and thus be blocked. In the worst case, another large job will be blocked, in which case J has paid Δ for itself, $\Delta - 1$ for the pseudo-blocked job, and Δ for a blocked job. Thus it has paid at most $2\Delta + \Delta - 1 \leq 2\Delta + \lceil\Delta\rceil - 1$. This completes our proof. ∎

For jobs of two distinct lengths, we are able to match the lower bound of Theorem 15 when $1 \leq \omega < \Delta$. Our proof will be similar to the proof of the upper bound for unit-length jobs in Section 5.1, but considerably more complicated. Without loss of generality, we prove the competitiveness by analyzing a single *busy period* in σ. Given both σ and σ^*, we construct σ' as follows. We filter through the GREEDY-TWOLENGTHS schedule σ from beginning to end, and at any point t where σ starts a *large* job J that was never run in σ^*, we look at the set $\mathcal{A}$ of all *large* jobs from σ^* that are currently available. If $\mathcal{A}$ is non-empty, we throw away J for good, and replace the spot in the schedule with that member J_a of $\mathcal{A}$ with the earliest expiration. If J_a was originally scheduled in σ at a later time, then we leave the abandoned spot empty for now and allow it to filled later, if possible, as we filter that part of the original schedule. We call σ' the resulting schedule at the end of this process. Notice that by construction, $|\sigma'| \leq |\sigma|$. We immediately have the following lemma.

Lemma 7 *For any time t at which σ originally started a large job, if large job $J_k \in \sigma^* - \sigma'$ is available to σ', then σ' must start some large job $J_i \in \sigma^*$ at time t, with $x_i \leq x_k$.*

The proof of this lemma is omitted, as it is identical to that of Lemma 1, with the additional fact that J_i must be large as the GREEDY-TWOLENGTHS algorithm would never choose to start a small job when a large job was available.

Theorem 8 *For $\omega \geq 1$, the* GREEDY-TWOLENGTHS *algorithm is $\left(1 + \frac{\lceil\Delta\rceil}{\Delta}\right)$-competitive.*

Proof: Recalling that $|\sigma'| \leq |\sigma|$, we provide a direct charging scheme in which each job $J \in \sigma'$ is allowed to distribute up to $\left(1 + \frac{\lceil\Delta\rceil}{\Delta}\right)|J|$, and each job $J^* \in \sigma^*$ receives at least $|J^*|$ units. Given job J which runs in σ' at time $J^{\sigma'}$, we assign costs as follows:

- $|J|$ is assigned to itself, if this job also run in σ^*.
- 1 is assigned to any *small* job $J^* \in \sigma^*$ which is blocked or covered by J.
- If J has not yet paid out $\left(1 + \frac{\lceil\Delta\rceil}{\Delta}\right)|J|$, it pays all of the excess to the *large* job of $\sigma^* - \sigma'$ with the smallest expiration time greater than or equal to $J^{\sigma'}$.

It is fairly clear that each job of σ' has assigned at most $\left(1 + \frac{\lceil\Delta\rceil}{\Delta}\right)|J|$. The third rule cannot possibly violate this, therefore we only worry about a job paying for itself and any small blocked or covered jobs. A small job can block at most one other small job, and thus pays at most 1 to itself and 1 to a blocked job by the first two rules. A large job can block or cover at most $\lceil\Delta\rceil$ small jobs, and thus pays at most Δ to itself and $\lceil\Delta\rceil$ to blocked jobs, for a total of $\Delta + \lceil\Delta\rceil = \left(1 + \frac{\lceil\Delta\rceil}{\Delta}\right)\Delta$.

The more difficult fact to prove is that every job $J^* \in \sigma^*$ is paid at least $|J^*|$. Any job that ran in σ' will have paid for itself. Also, any *small* job not in σ' must have been blocked by some job, and that job will have paid it 1. Therefore, we must only show that any large jobs in $\sigma^* - \sigma'$ were paid enough. This payment will come from the third assignment rule. Let J^* be a large job in $\sigma^* - \sigma'$, with expiration time x. We define t to be the latest time at which a job starts in σ', even though J^* was available, and we assume job J_t is the job which ran. Note that because $\omega \geq 1$, the window of availability for J^* is at least Δ, and so no single job can block the entire window. Therefore, we are assured that such a time t exists. We have that $t \leq x < t + \Delta$. Our construction assures us, through Lemma 7, that J_t is a large job, also in σ^*, and with expiration time at most x. Now we define $s \leq t$ to be the minimum start time in σ' such that σ' is never idle in the interval $[s, t]$, and that all jobs started during the interval $[s, t]$ are large, also in σ^*, and have expiration time at most x. We denote as R the set of all jobs that σ' begins during $[s, t]$, together with the job J^*. Lemma 7 assures us that every job of R is contained in σ^* and has an expiration time of at most x. Note that it must be the case that $t - s = k\Delta$ for some integer $k \geq 0$, and so the set R contains $k + 2$ jobs (J^* as well as the $k + 1$ jobs that start during the interval).

We break our proof into three cases depending on what was running in σ immediately prior to time s. Either the resource was idle, running a small job, or running a large job. Actually, we begin by immediately showing that it is not possible that the resource was idle immediately prior to s. Assume that it was idle. In this case, it must be that all jobs of R arrived on or after time s, as our greedy algorithm would not have been idle if any of these jobs had already arrived, and our construction of σ' would not have left an empty spot if a job of R could have been placed there. Recall that all jobs of R expire on or before time $x < t + \Delta$, and that all jobs of R are scheduled in σ^*. However this is not possible, as all jobs of R must be scheduled in σ^* to be completely contained during the interval $[s, x + \Delta] \subset [s, t + 2\Delta)$, yet strictly less than $\frac{t+2\Delta-s}{\Delta} = k + 2$ jobs can be completely contained in this half-open interval.

So there must be some job J_r, small or large, which runs immediately prior to time s, beginning at time $s - |J_r|$. Now we show that J^* will have been paid enough by the third rule, using the excess from jobs in R along with excess from job J_r. We do this in two steps, first showing that the amount of this excess is at least equal to Δ, an then showing that all of this excess will indeed be paid to job J^*.

First, we consider the case where J_r is a small job. Since this job started at time $s - 1$, there must not have been any available large job, and therefore it must be that all jobs of R arrive strictly after time $s - 1$. As we know, all jobs of R expire on or before time $x < t + \Delta$, and yet σ^* runs all of these jobs. Therefore, all $k + 2$ jobs of R must be entirely contained during the interval $(s - 1, t + 2\Delta)$ of σ^*, with length less than $t + 2\Delta - (s - 1) = (k + 2)\Delta + 1$. Therefore, there is strictly less than 1 unit of time remaining in the interval, and so at most 1 small job can also begin during this interval. If we consider the payments of J_r and jobs of R, we see that in addition to possibly paying for themselves, they are allowed to pay $\frac{\lceil\Delta\rceil}{\Delta}(1 + (k + 1)\Delta) \geq \frac{\lceil\Delta\rceil}{\Delta}(1 + \Delta) \geq 1 + \Delta$. Since at most 1 unit is paid to blocked small jobs, there are at least Δ units of remaining excess. We claim that all of this excess is in fact paid to J^*. If any of this excess were paid to some other large job, call it J_o, then this job must be in $\sigma^* - \sigma'$ and not in the set R. Furthermore this job must expire sometime between $s - 1$ and x, as it got paid the excess, ahead of J^*, by a job which started during this period. It must be that J_o arrived strictly after time $s - 1$, since σ' chose to start a small job at $s - 1$. But now, in addition to the set R, we have another job which arrives on or after $s - 1$, expires on or before $x < t + \Delta$, and yet runs in σ^*. This means that $k + 3$ large jobs must start in the interval $[s - 1, t + \Delta)$, which is impossible. Therefore, we are sure that any such excess is in fact paid to J^*, and so we have seen that J^* is paid at least Δ.

Finally, we consider the case where J_r is a large job. This job starts at time $s - \Delta$. We claim that all jobs of R must arrive strictly after time $s - \Delta$, based on Lemma 7 combined with our definition of the choice of s. As we know, all jobs of R expire on or before time $x < t + \Delta$, and yet σ^* runs all of these jobs. Therefore, all $k + 2$ jobs of R must be entirely contained during the interval $(s - \Delta, t + 2\Delta)$ of σ^*, with length less than $t + 2\Delta - (s - \Delta) = (k + 3)\Delta$. Therefore, there is strictly less than Δ units of time remaining in the interval, and so at most Δ small job can also begin during this interval. If we consider the payments of J_r and

jobs of R, we see that in addition to possibly paying for themselves, they are allowed to pay $\frac{\lceil\Delta\rceil}{\Delta}((k+2)\Delta) \geq \frac{\lceil\Delta\rceil}{\Delta}(2\Delta) \geq \lceil\Delta\rceil + \Delta$. Since at most $\lceil\Delta\rceil$ units are paid to blocked small jobs, there are at least Δ units of remaining excess. We claim that all of this excess is in fact paid to J^*. Assume, instead, that some of this excess were paid to some other large job, call it $J_o \in \sigma^* - \sigma'$, with expiration $x_o \leq x$. If this job arrives strictly after time $s - \Delta$, then we produce a contradiction by the identical argument as the case where J_r was small. Therefore, we assume that job J_r arrived before $s - \Delta$. However, now we ask why this job was not chosen to begin at $s - \Delta$. According to Lemma 7, it must be that the job which did start then was large, contained in σ, and has expiration less than or equal to $x_o \leq x$. But this exactly contradicts our original definition of s, as the interval $[s - \Delta, t]$ would have satisfied the property we used to define s. Therefore, we are sure that any such excess is in fact paid to J^*, and so we have seen that J^* is paid at least Δ, completing our proof. ∎

Corollary 9 *If $\omega > 0$,*

$$\hat{D}_2(\omega) \leq 1 + \max\left(\frac{\lceil\omega\rceil + 1}{\lceil\omega\rceil}, \frac{\lfloor\omega\rfloor + 1}{\omega}\right).$$

Proof: For a fixed ω, we prove this bound on $D(\omega, \Delta)$ over all possible values of Δ.

- When $0 < \omega < \frac{1}{\Delta} < 1$, we can apply Theorem 5, which assures us that $D_2(\omega, \Delta) \leq \max(2+\frac{1}{\Delta}, 1+\Delta)$. Notice that $\max(2+\frac{1}{\Delta}, 1+\Delta) \leq \max(3, 1+\frac{1}{\omega}) = 1+\max(2, \frac{1}{\omega})$. Since $\lfloor\omega\rfloor = 0$ and $\lceil\omega\rceil = 1$, it is exactly the case that $1+\max(2, \frac{1}{\omega}) = 1+\max(\frac{\lceil\omega\rceil+1}{\lceil\omega\rceil}, \frac{\lfloor\omega\rfloor+1}{\omega})$.

- When $\frac{1}{\Delta} \leq \omega < 1$, we can apply Theorem 6, which assures us that $D_2(\omega, \Delta) \leq 2 + \frac{\lceil\Delta\rceil-1}{\Delta}$. Notice that $2 + \frac{\lceil\Delta\rceil-1}{\Delta} \leq 3 = 1 + \frac{\lceil\omega\rceil+1}{\lceil\omega\rceil}$, and so $D_2(\omega, \Delta) \leq 1 + \frac{\lceil\omega\rceil+1}{\lceil\omega\rceil} \leq 1 + \max\left(\frac{\lceil\omega\rceil+1}{\lceil\omega\rceil}, \frac{\lfloor\omega\rfloor+1}{\omega}\right)$.

- When $1 \leq \omega < \Delta$, we can apply Theorem 8, which assures us that $D_2(\omega, \Delta) \leq 1 + \frac{\lceil\Delta\rceil}{\Delta}$. If it happens that $\lceil\omega\rceil \leq \Delta$, then $1 + \frac{\lceil\Delta\rceil}{\Delta} = 2 + \frac{\lceil\Delta\rceil-\Delta}{\Delta} < 2 + \frac{1}{\Delta} \leq 2 + \frac{1}{\lceil\omega\rceil} = 1 + \frac{\lceil\omega\rceil+1}{\lceil\omega\rceil}$, satisfying our bound. Otherwise, we have that $\omega < \Delta < \lceil\omega\rceil = \lceil\Delta\rceil = \lfloor\omega\rfloor + 1$. In this case, $1 + \frac{\lceil\Delta\rceil}{\Delta} = 1 + \frac{\lfloor\omega\rfloor+1}{\Delta} < 1 + \frac{\lfloor\omega\rfloor+1}{\omega}$.

- When $\Delta \leq \omega$, we can apply the more general Theorem 11, proven in the next section, which assures us that $D_2(\omega, \Delta) \leq D(\omega, \Delta) \leq 2 + \frac{1}{\lceil\omega\rceil} = 1 + \frac{\lceil\omega\rceil+1}{\lceil\omega\rceil}$.

We have proven the upper bound for every relevant combination of ω and Δ, and so the proof is complete. ∎

5.3 Arbitrary Length Jobs

In this section, we consider the case where jobs can have arbitrary lengths in the range $[1, \Delta]$, and where the minimum allowable delay for any job of length $|J|$ is equal to $\omega|J|$.

Theorem 10 *For $\omega > 0$ and all Δ,*

$$D(\omega, \Delta) \leq 2 + \frac{1}{\omega}.$$

Proof: We prove this upper bound by showing that any *greedy-type* algorithm achieves such a competitive ratio. Our proof structure is similar to the analysis of GREEDY in Section 5.1. Without loss of generality, we prove the competitiveness by analyzing a single *busy period* in σ. Again, we let $\mathcal{L}$ denote the jobs in σ^* scheduled to begin on or after time e (i.e. 'late'), and $\mathcal{N}$ denote the jobs in σ that were not members of $\mathcal{L}$. In a slightly different manner we now let $\mathcal{P}$ denote jobs in σ^* scheduled to end on or before time e (i.e. 'prompt'), and finally we let job J be the single job of σ^*, if one exits, which begins strictly before time e yet ends strictly after time e. We let $L = \sum_{J_i \in \mathcal{L}} |J_i|$, and define values N and P analogously. By definition, we have that $|\sigma^*| = P + |J| + L$, an again we have that $|\sigma| \geq P$ and also that $|\sigma| = N + L$, as we know that $\mathcal{L} \subseteq \sigma$. From this, we see that $P + N + L \leq 2|\sigma|$. Adding $|J| - N$ to each side, we see that $|\sigma^*| = P + |J| + L \leq 2|\sigma| + (|J| - n)$. Rearranging, we have that $\frac{|\sigma^*|}{|\sigma|} \leq 2 + \frac{|J|-N}{|\sigma|}$.

Now, if $J \in \mathcal{N}$ or if J does not exist, then we see that $|J| - N \leq 0$, and therefore it must be that $\frac{|\sigma^*|}{|\sigma|} \leq 2$. Otherwise, we have that J exists and that J did not run in σ. Now we take advantage of the fact that J must have had a window of availability of at least $\omega|J|$. Since it never ran in σ, and σ was produced by a greedy-type algorithm, then σ must have been running other jobs throughout the window and so $|\sigma| \geq \omega|J|$. But in this case, $\frac{|J|-N}{|\sigma|} \leq \frac{|J|-N}{\omega|J|} \leq \frac{|J|}{\omega|J|} = \frac{1}{\omega}$. Therefore, our competitiveness is at most $2 + \frac{1}{\omega}$. ∎

Theorem 11 *For $\omega \geq \Delta$,*

$$D(\omega, \Delta) \leq 2 + \frac{1}{\lceil \omega \rceil}.$$

Proof: We prove this upper bound by showing that the GREEDY algorithm, which always schedules the available job with earliest expiration time achieves such a competitive ratio. Just as in the proof of Theorem 10, we analyze a signel busy period of σ, and we see that $\frac{|\sigma^*|}{|\sigma|} \leq 2 + \frac{|J|-N}{|\sigma|}$.

Again, if $J \in N$ or if J does not exist, we immediately have that $\frac{|J|-N}{|\sigma|} \leq 0$. Otherwise, we claim that $|\sigma| \geq \omega|J|$ and also that $N \geq 1$. Both of these facts are again based on the fact that J did not run in σ, although it had a window of availability of at least $\omega|J|$. Since J did not run in σ, it must be that σ was running other jobs throughout the window, and so $|\sigma| \geq \omega|J|$. Furthermore, since $\omega|J| \geq \Delta|J| \geq \Delta$, there must be at least one point at which σ began running a job while J was available. Because our algorithm chooses to run a

job with earliest expiration, the job which was run has an expiration time at least as early as J. Since J was missed by σ, we know that its expiration time must be earlier than e. Therefore the job chosen over J cannot be a late job, rather it is a job of $\mathcal{N}$, and as every job has length at least 1, we have that $N \geq 1$.

Now we are able to prove our bound. Because $|\sigma| \geq \omega|J|$ and $N \geq 1$, we see that $\frac{|J|-N}{|\sigma|} \leq \frac{|J|-1}{|\sigma|} \leq \frac{|J|-1}{\omega|J|}$. Finally, since $|J| \leq \Delta \leq \omega$, it must be that $\frac{|J|-1}{\omega|J|} \leq \frac{\omega-1}{\omega^2}$. Since $\frac{1}{\omega+1} - \frac{\omega-1}{\omega^2} = \frac{1}{\omega^2(\omega+1)} > 0$, we see that $\frac{\omega-1}{\omega^2} < \frac{1}{\omega+1} < \frac{1}{\lceil\omega\rceil}$. In summary, $\frac{\sigma^*}{\sigma} \leq 2 + \frac{|J|-1}{\omega|J|} \leq 2 + \frac{1}{\lceil\omega\rceil}$. ∎

6 Lower Bounds

Our lower bounds have the following basic form. In each construction, the online algorithm is given a single initial job, with a large delay, and it will have to decide when to run it. As this job may be the only one to arrive, any algorithm will have to run this job with non-zero probability at some point if it is still the only job to arrive. Otherwise such an algorithm would have a gain of zero, where the optimal algorithm has positive gain. For deterministic lower bounds, we know the exact time at which an algorithm chooses to run this initial job, and we can build the rest of our construction to begin after this time. For randomized lower bounds, we have to prepare two different constructions, one for which the best action is to run the initial job as soon as it arrives, and a second for which the best action is to wait, and we show that an oblivious adversary can always choose one of the scenarios that will be bad for a given randomized algorithm.

We consider the following three general constructions.

1. The initial job has unit length, and following it will be $z \geq 1$ large jobs, such that any algorithm that initially runs the first job we be forced to miss at least one of the z jobs.

2. The initial job is large, and following it will be k small jobs, such that any algorithm that initially runs the first job we be forced to miss *all* k small jobs.

3. There is an initial job, and following it will be k small jobs and z large jobs, such that any algorithm that initially runs the first job we be forced to miss *all* k small jobs and miss at least one of the z large jobs.

Our choice of which construction to use, and the exact parameters chosen depend on the exact problem setting being considered.

6.1 Unit-Length Jobs

For this section, we concern ourselves only with unit-length jobs ($l_i = 1$), and thus all jobs have a minimum wait time of at least ω. We use the first type of general construction, except that we only use jobs of unit length.

Figure 2: The left drawing shows arrivals and deadlines for instance $\mathcal{S}_1$, in which it is necessary to schedule job J_1 last. The right drawing shows instance $\mathcal{S}_2$, for which it is necessary to schedule job J_1 at time $t = 0$.

Theorem 12 *For all ω,*

$$D_1(\omega) \geq 1 + \frac{1}{\lfloor \omega \rfloor + 1} \qquad \textit{and} \qquad R_1(\omega) \geq 1 + \frac{1}{2\lfloor \omega \rfloor + 3}.$$

Proof: Let $z = \lfloor \omega \rfloor + 1$, and thus $\omega = z - 2\epsilon$ for some $0 < 1\epsilon \leq 1$. Consider the following two scenarios, consisting of $z + 1$ jobs with minimum wait ω. In the first scenario $\mathcal{S}_1$, we let job $J_1 = \langle 0, 1, z + \epsilon \rangle$, and thus with deadline $z + 1 + \epsilon$. We introduce z other jobs each with parameters $\langle \epsilon, 1, z - 2\epsilon \rangle$, and thus with deadlines at $z + 1 - \epsilon$. For this input, it is possible to schedule all $z + 1$ jobs, by running the new jobs from time $[\epsilon, z + \epsilon)$ followed by job J_1 from $[z + \epsilon, z + 1 + \epsilon)$. However, any other schedule runs at most z of the jobs.

Our second scenario, $\mathcal{S}_2$, consists of the same first job J_1, however we now introduce z other jobs each with parameters $\langle 2\epsilon, 1, z - 2\epsilon \rangle$, and thus with deadlines at $z + 1$. For this input, it is possible to schedule all $z + 1$ jobs, by running job J_1 at time $[0, 1)$, followed by the new jobs from time $[1, z+1)$. Again, any other schedule runs at most z of the jobs. Both of these scenarios are shown in Figure 2.

Notice that at time $t = 0$, both of these inputs look identical to an online algorithm. Given any deterministic algorithm A, we consider whether that algorithm schedules job J_1 at time $t = 0$. In either case, in one of the two scenarios above, it will only schedule z tasks, whereas the optimal algorithm will schedule $z + 1$ in either case. Therefore A's competitive ratio must be at least $\frac{z+1}{z} = 1 + \frac{1}{z} = 1 + \frac{1}{\lfloor \omega \rfloor + 1}$.

Given any randomized algorithm A, there must be some fixed probability p that it chooses to run job J_1 at time $t = 0$ for either scenario. If $p > \frac{1}{2}$, we consider scenario $\mathcal{S}_1$, otherwise we consider scenario $\mathcal{S}_2$. In either case, we are assured that at least $1/2$ of the time, algorithm A will fail to schedule at least one job, whereas the optimal offline algorithm will always schedule all $z+1$ jobs. Therefore, algorithm A has an expected gain of at most $\frac{1}{2} \cdot z + \frac{1}{2} \cdot (z+1) = z + \frac{1}{2}$. The competitive ratio must be at least $\frac{z+1}{z+\frac{1}{2}} = 1 + \frac{1}{2z+1} = 1 + \frac{1}{2\lfloor \omega \rfloor + 3}$. ∎

6.2 Jobs of Two Distinct Lengths

In this section, we consider the case where all jobs are either of length 1 or length $\Delta > 1$, and where the delay for any job of length $|J|$ is equal to at least $\omega|J|$. We refer to jobs of length 1 as "small" jobs, and jobs of length Δ as "large" jobs.

Theorem 13 *If* $\omega < \frac{1}{\Delta}$,

$$D_2(\omega, \Delta) \geq 1 + \Delta.$$

Proof: This construction is of the first general form. Let $\epsilon = \frac{1-\omega\Delta}{2} > 0$. We first consider the behavior of an algorithm A when faced with job $J_1 = \langle 0, 1, 1 + \Delta + \epsilon \rangle$. If A is deterministic, assume it begins running job J_1 at time t. Now we consider an additional job $J_2 = \langle t + \epsilon, \Delta, 1 - 2\epsilon \rangle$.

Notice that the delay $1 - 2\epsilon = \omega\Delta$, and thus the minimum delay requirement is met for J_2. The expiration time for J_2 is equal to $t + 1 - \epsilon$. Since A runs job J_1 from $[t, t + 1)$, it misses J_2, and thus has gain at most 1. However, the optimal schedule can achieve both jobs, and thus have gain $1 + \Delta$. This is done by running the large job during $[t + \epsilon, t + \epsilon + \Delta)$. Depending on the value of t, job J_1 always fits either immediately before or after this range. Therefore the competitive ratio for any such algorithm A is no better than $\frac{1+\Delta}{1} = 1 + \Delta$. ∎

Theorem 14 *If* $\omega < 1$,

$$D_2(\omega, \Delta) \geq 2 + \frac{\lceil \Delta \rceil - 1}{\Delta}.$$

Proof: The construction is of the third general form. Let $\epsilon = \frac{1}{2} \min(\Delta + 1 - \lceil \Delta \rceil, (1 - \omega)\Delta) > 0$. We first consider the behavior of an algorithm A when faced with job $J_1 = \langle 0, \Delta, 3\Delta - \epsilon \rangle$. If A is deterministic, assume it begins running job J_1 at time t. Now we consider an additional large job $J_2 = \langle t + \epsilon, \Delta, \Delta - 2\epsilon \rangle$, as well as $\lceil \Delta \rceil - 1$ additional small jobs, each with parameters $\langle t + \epsilon, 1, \Delta - 2\epsilon \rangle$.

Notice that the delay of all jobs $\Delta - 2\epsilon \geq \omega\Delta$, and thus the minimum delay requirement is met for all of our jobs. The expiration time for all new jobs is equal to $t + \Delta - \epsilon$, and so A will miss them all, as it runs J_1 from $[t, t + \Delta)$. However, the optimal schedule achieves all jobs, and thus has gain $2\Delta + \lceil \Delta \rceil - 1$. This is done by running the small jobs from $[t + \epsilon, t + \epsilon + \lceil \Delta \rceil - 1)$ and the large job from $[t + \Delta - \epsilon, t + 2\Delta - \epsilon)$. Depending on the value of t, job J_1 always fits either immediately before or after the other jobs. Therefore, the competitive ratio for any such algorithm A is no better than $\frac{2\Delta + \lceil \Delta \rceil - 1}{\Delta} = 2 + \frac{\lceil \Delta \rceil - 1}{\Delta}$. ∎

Theorem 15 *If* $\omega < \Delta$,

$$D_2(\omega, \Delta) \geq 1 + \frac{\lceil \Delta \rceil}{\Delta}.$$

Proof: This construction is of the second general form. Let $\epsilon = \frac{1}{2} \min(\Delta - \omega, \Delta + 1 - \lceil \Delta \rceil) > 0$. We first consider the behavior of an algorithm A when faced with job $J_1 = \langle 0, \Delta, \max(\omega\Delta, \Delta + \lceil \Delta \rceil + \epsilon) \rangle$. If A is deterministic, assume it begins running job J_1 at time t. Now we consider an additional $\lceil \Delta \rceil$ small jobs, each with parameters $\langle t + \epsilon, 1, \Delta - 2\epsilon \rangle$.

Notice that the delay of the small jobs, $\Delta - 2\epsilon \geq \omega$, and thus the minimum delay requirement is met for all of our jobs. The expiration time for each of the new small jobs is equal to $t + \Delta - \epsilon$, and thus all of these jobs are lost by A which runs J_1 from $[t, t + \Delta)$.

Algorithm A has gain at most Δ, however the optimal schedule can achieve all jobs, and thus have gain $\Delta + \lceil\Delta\rceil$. This is done by running the small jobs from $[t+\epsilon, t+\epsilon+\lceil\Delta\rceil)$. Notice that the last of these jobs began exactly at $t+\epsilon+\lceil\Delta\rceil-1 \leq t+\Delta-\epsilon$. Depending on the value of t, job J_1 always fits either immediately before or after the other jobs. Therefore the competitive ratio for any such algorithm A is no better than $\frac{\Delta+\lceil\Delta\rceil}{\Delta} = 1+\frac{\lceil\Delta\rceil}{\Delta}$. ∎

Corollary 16 $\hat{D}_2(0) = \infty$.

Proof: We can choose Δ arbitrarily large while Theorem 13 still applies, giving a lower bound of $1+\Delta$ for arbitrarily large Δ. ∎

Corollary 17 *For all* $\omega > 0$,

$$\hat{D}_2(\omega) \geq 1 + \max\left(\frac{\lceil\omega\rceil+1}{\lceil\omega\rceil}, \frac{\lfloor\omega\rfloor+1}{\omega}\right).$$

Proof: If $0 < \omega < 1$, the claim reduces to proving that $\hat{D}_2(\omega) \geq \max(3, 1+\frac{1}{\omega})$. First, let ϵ be arbitrarily small such that $0 < \epsilon < 1-\omega$, and consider $\Delta = \frac{1-\epsilon}{\omega} > 1$, for an arbitrary small value of $\epsilon > 0$, then we have $\omega < \frac{1}{\Delta}$, and so Theorem 13 states that $D_2(\omega, \Delta) \geq 1+\Delta$. Notice that $1+\Delta = 1+\frac{1}{\omega}-\frac{\epsilon}{\omega}$. Since we can choose ϵ arbitrarily small, this lower bound approaches $1+\frac{1}{\omega}$, and so $\hat{D}_2(\omega) \geq 1+\frac{1}{\omega}$. Alternatively, we could consider arbitrarily small $\epsilon > 0$, and let $\Delta = 1+\epsilon$. In this case, Theorem 14 tells us that $D_2(\omega, \Delta) \geq 2+\frac{\lceil\Delta\rceil-1}{\Delta} = 2+\frac{2}{1+\epsilon}$. Since ϵ can be made arbitrarily small, this can be made arbitrarily close to 3 and thus $\hat{D}_2(\omega) \geq 3$. Together, these two bounds prove the result for $0 < \omega < 1$.

When $\omega \geq 1$, we again consider two possibilities. In the first, we let $\Delta = \lceil\omega\rceil + \epsilon$ for arbitrarily small $\epsilon > 0$. Theorem 15 assures us that

$$D_2(\omega, \Delta) \geq 1 + \frac{\lceil\Delta\rceil}{\Delta} = 1 + \frac{\lceil\lceil\omega\rceil+\epsilon\rceil}{\lceil\omega\rceil+\epsilon} = 1 + \frac{\lceil\omega\rceil+1}{\lceil\omega\rceil+\epsilon}.$$

Since ϵ can be chosen arbitrarily small, this can be made arbitrarily close to $1+\frac{\lceil\omega\rceil+1}{\lceil\omega\rceil}$.

Secondly, we let $\Delta = \omega + \epsilon$ for arbitrarily small $\epsilon > 0$. Theorem 15 assures us that

$$D_2(\omega, \Delta) \geq 1 + \frac{\lceil\Delta\rceil}{\Delta} = 1 + \frac{\lceil\omega+\epsilon\rceil}{\omega+\epsilon} = 1 + \frac{\lfloor\omega\rfloor+1}{\omega+\epsilon}.$$

Since ϵ can be chosen arbitrarily small, this can be made arbitrarily close to $1+\frac{\lfloor\omega\rfloor+1}{\omega}$. Combining these two possibilities, we see that $\hat{D}_2(\omega) \geq 1+\max(\frac{\lceil\omega\rceil+1}{\lceil\omega\rceil}, \frac{\lfloor\omega\rfloor+1}{\omega})$. ∎

6.3 Arbitrary Length Jobs

In this section, we consider the case where jobs can have arbitrary lengths in the range $[1, \Delta]$, and where the minimum allowable delay for any job of length $|J|$ is equal to $\omega|J|$. It is worth noting that all of our lower bounds for $\hat{D}(\omega)$ involve at most three distinct job lengths.

Theorem 18 *If* $\omega < \frac{1}{\Delta}$,

$$D(\omega, \Delta) \geq 2 + \Delta.$$

Proof: The construction is of the third general form. Let ϵ be an arbitrarily small constant $\epsilon > 0$. We first consider the behavior of an algorithm A when faced with job $J_1 = \langle 0, 1 + 2\epsilon, 1 + \Delta + \epsilon \rangle$. If A is deterministic, assume it begins running job J_1 at time t. Now we consider two additional jobs $J_2 = \langle t + \epsilon, 1, 1 \rangle$ and $J_3 = \langle t + \epsilon, \Delta, 1 \rangle$.

Since $\omega < \frac{1}{\Delta}$, the delay $1 < \omega\Delta$ satisfies the minimum delay requirement for both jobs. The expiration time for both new jobs is equal to $t + 1 + \epsilon$. Since A runs job J_1 from $[t, t + 1 + 2\epsilon)$, it misses both J_2 and J_3, and thus has gain at most $1 + 2\epsilon$. However, the optimal schedule achieves all jobs, and thus have gain $\Delta + 2 + 2\epsilon$. This is done by running J_2 from $[t+\epsilon, t+1+\epsilon)$ and J_3 from $[t+1+\epsilon, t+1+\Delta+\epsilon)$. Depending on the value of t, job J_1 always fits either immediately before or after this range. Therefore the competitive ratio for any such algorithm A is no better than $\frac{\Delta+2+\epsilon}{1+2\epsilon}$. Since ϵ can be chosen to be arbitrarily small, this lower bound can be made arbitrarily close to $2 + \Delta$. Therefore, it cannot be the case that any deterministic algorithm has a competitive ratio for $c < 2 + \Delta$. ∎

Theorem 19 *If* $\frac{1}{\Delta} \leq \omega < 1$,

$$D(\omega, \Delta) \geq 2 + \frac{1}{\omega}.$$

Proof: The construction is of the third general form. Notice that for $\frac{1}{\Delta} \leq \omega < 1$, we have that $1 \leq \lfloor \omega\Delta \rfloor < \frac{\lfloor \omega\Delta \rfloor}{\omega} \leq \Delta$. Let ϵ be an arbitrarily small constant such that $0 < \epsilon \leq \frac{\Delta - \lfloor \omega\Delta \rfloor}{2}$. Let job $J_1 = \langle 0, \lfloor \omega\Delta \rfloor + 2\epsilon, \left(2 + \frac{1}{\omega}\right) \lfloor \omega\Delta \rfloor + 3\epsilon \rangle$, and consider the behavior of an algorithm A when faced with job J_1. If A is deterministic, assume it begins running job J_1 at time t. Now we consider an additional job $J_2 = \langle t + \epsilon, \frac{\lfloor \omega\Delta \rfloor}{\omega}, \lfloor \omega\Delta \rfloor \rangle$, as well as $\lfloor \omega\Delta \rfloor$ additional unit-length jobs, $\langle t + \epsilon, 1, \lfloor \omega\Delta \rfloor \rangle$.

The expiration time for all new jobs is equal to $t + \lfloor \omega\Delta \rfloor + \epsilon$, and so A will miss them all, as it runs J_1 from $[t, t + \lfloor \omega\Delta \rfloor + 2\epsilon)$. Therefore, the gain of A is at most $\lfloor \omega\Delta \rfloor + 2\epsilon$, whereas the optimal schedule can achieve all jobs for a gain of $\left(2 + \frac{1}{\omega}\right) \lfloor \omega\Delta \rfloor + 2\epsilon$, by running the small jobs from $[t + \epsilon, t + \lfloor \omega\Delta \rfloor + \epsilon)$, and J_2 from $[t + \lfloor \omega\Delta \rfloor + \epsilon, t + (1 + \frac{1}{\omega}) \lfloor \omega\Delta \rfloor + \epsilon)$. Depending on the value of t, job J_1 always fits either immediately before or after this range. Therefore the competitive ratio for any such algorithm A is no better than $\frac{\left(2+\frac{1}{\omega}\right)\lfloor \omega\Delta \rfloor + 2\epsilon}{\lfloor \omega\Delta \rfloor + 2\epsilon}$. Since ϵ can be chosen to be arbitrarily small, this lower bound can be made arbitrarily close to $2 + \frac{1}{\omega}$. ∎

Theorem 20 *If* $1 \leq \omega \leq \lceil \Delta \rceil - 1$,

$$D(\omega, \Delta) \geq 2 + \frac{1}{\omega}.$$

Proof: The construction is also of the third general form. Let ϵ be an arbitrarily small constant such that $0 < \epsilon \leq \frac{\Delta+1-\lceil\Delta\rceil}{2}$. Let integer $K = \lceil \Delta \rceil - 1$. Consider the behavior of an algorithm A when faced with job $J_1 = \langle 0, K + 2\epsilon, \left(2 + \frac{1}{\omega}\right) K + 3\epsilon \rangle$. If A is deterministic, assume it begins running job J_1 at time t. Now we consider an additional job $J_2 = \langle t + \epsilon, \frac{K}{\omega}, K \rangle$, as well as K additional unit-length jobs, $\langle t + \epsilon, 1, K \rangle$. Notice that $|J_2| = \frac{K}{\omega}$ is such that $1 \leq \frac{K}{\omega} \leq \Delta$.

The expiration time for all new jobs is equal to $t + K + \epsilon$, and so A will miss them all, as it runs J_1 from $[t, t + K + 2\epsilon)$. Therefore, the gain of A is at most $K + 2\epsilon$, whereas the optimal schedule can achieve all jobs for a gain of $K\left(2 + \frac{1}{\omega}\right) + 2\epsilon$, by running the small jobs from $[t + \epsilon, t + K + \epsilon)$, and J_2 from $[t + K + \epsilon, t + K\left(1 + \frac{1}{\omega}\right) + \epsilon)$. Depending on the value of t, job J_1 always fits either immediately before or after this range. Therefore the competitive ratio for any such algorithm A is no better than $\frac{K\left(2+\frac{1}{\omega}\right)+2\epsilon}{K+2\epsilon}$. Since ϵ can be chosen to be arbitrarily small, this lower bound can be made arbitrarily close to $2 + \frac{1}{\omega}$. ∎

Corollary 21 $\hat{D}(0) = \infty$.

Proof: When $\omega = 0$, we can let Δ be arbitrarily large while Theorem 18 still applies, giving a lower bound of $2 + \Delta$ for arbitrarily large Δ. ∎

Corollary 22 *For* $\omega > 0$,

$$\hat{D}(\omega) \geq 2 + \frac{1}{\omega}.$$

Proof: For any value of ω, we can always apply either Theorem 19 or 20, by choosing a large enough value of Δ. ∎

7 Concluding Remarks

In this paper, we provide tight bounds for the competitiveness of deterministic algorithms in the scheduling model where jobs are required to accept minimum delays proportional to their jobs lengths. We show the following results,

- When arbitrary job lengths are allowed, we give a $(2 + \frac{1}{\omega})$-competitive deterministic algorithm, and we provide a lower bound showing that this is the best possible deterministic results for all values of $\omega > 0$.
- When all jobs have the same length, we show that the GREEDY algorithm is $(1 + \frac{1}{\lfloor\omega\rfloor+1})$-competitive for all values of $\omega \geq 0$, and we show that this is the best possible result for deterministic algorithms.

- When all jobs have one of two distinct lengths, we provide a tight bound for the deterministic competitiveness when $\omega > 0$, equal to $1 + \max(\frac{\lceil\omega\rceil+1}{\lceil\omega\rceil}, \frac{\lfloor\omega\rfloor+1}{\omega})$.

There are several open questions. The first of which is to better understand whether randomization can be used to improve on the deterministic results. In this paper, we have set up the notation for the study of randomized online algorithms, but for the most part our results have involved deterministic algorithms. As a first step, we point towards a remaining open question of [6] concerning the case of unit-length jobs with no minimum delay. They show that the GREEDY algorithm is a deterministic 2-competitive algorithm, and yet the strongest randomized lower bounds still allows for the possibility for a 4/3-competitive algorithm. We feel that closing the gap for this case is a necessary precursor to improving on the results for values of $\omega > 0$. Additionally, some gaps remain in the appendix, regarding the exact deterministic competitiveness when parameterized by both ω and Δ.

This scheduling model may be generalized in several ways. The problem can be formulated in a multi-processor setting, where more than one channel is available to the scheduler. Also, as is pointed out in [6], we could add an additional parameter for each job specifying the *payoff* that the algorithm receives if that job is scheduled (in our model, the payoff is assumed to be exactly the length of the job). Finally, we ask whether a minimum delay requirement can be used to improve the competitiveness in other online scheduling problems.

Acknowledgments. The author would like to thank Sally Goldman and Subhash Suri for many insights into their previous work on this problem, and to thank Sanjeev Arora for several helpful conversations.

References

[1] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive online routing. In *Proc. 33rd Symp. on Found. Comput. Sci.*, pages 32–40, 1993.

[2] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosén. Competitive non-preemptive call control. In *Proc. 5th ACM Symp. on Discrete Algorithms*, pages 312–320, 1994.

[3] S. Baruah, G. Koren, B. Mishra, A. Ragunathan, L. Rosier, and D. Shasta. Online scheduling in the presence of overload. In *Proc. 31st Symp. on Found. Comput. Sci.*, pages 100–110, 1991.

[4] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Widgerson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994.

[5] A. Feldmann, B. Maggs, J. Sgall, D. Sleator, and A. Tomkins. Competitive analysis of call admission algorithms that allow delay. Technical Report CMU-CS-95-102, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1995.

[6] S. Goldman, J. Parwatikar, and S. Suri. On-line scheduling with hard deadlines. In *Proc. of the Workshop on Algorithms and Data Structures*, volume 1272 of *Lecture Notes in Computer Science*, pages 258–271. Springer-Verlag, 1997.

[7] A. Karlin, M. Manasse, L. Rudolph, and D. Sleator. Competitive snoopy paging. *Algorithmica*, 3(1):70–119, 1988.

[8] R. Lipton and A. Tomkins. Online interval scheduling. In *Proc. 5th ACM Symp. on Discrete Algorithms*, pages 302–311, 1994.

[9] D. Long and M. Thakur. Scheduling realtime disk transfers for continuous media applications. In *Proc. 12th IEEE Symp. on Mass Storage Systems*, pages 227–232, 1993.

[10] P. Raghavan and M. Snir. Memory versus randomization in on-line algorithms. *IBM Journal of Research and Development*, 38:683–707, 1994.

[11] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.

A Exact bounds parameterized by (ω, Δ)

In the main results of the paper, our goal was to better understand the bounds on competitiveness for fixed ω when we put no restrictions on Δ. For the sake of completeness, in this section we attempt to give good upper and lower bounds for the competitiveness of the problem when both ω and Δ are fixed parameters.

A.1 Deterministic Online Algorithms

Theorem 23 *For all ω and Δ, any* greedy-type *algorithm is $(2+\Delta)$-competitive.*

Proof: We use the identical charging scheme as Theorem 5, where every J job pays $|J|$ to itself, and pays $|J'|$ to any job J' which it blocks or covers. Again, it is clear that each job $J' \in \sigma^*$ was paid at least $|J'|$. In order to show that each job $J \in \sigma$ pays out at most $(2+\Delta)|J|$ units, we notice that the sum of the lengths of jobs covered by it is at most $|J|$, and that it can block at most one additional job. Therefore, it pays at most $|J|$ to itself, at most $|J|$ to covered jobs, and at most Δ to the one additional blocked job, totaling at most $2|J| + \Delta \leq 2|J| + \Delta|J| = |J|(2+\Delta)$. ∎

A.2 Lower Bounds

Corollary 24 *If $\omega < \frac{1}{\Delta}$, then*

$$D_2(\omega, \Delta) \geq \max(1+\Delta, 2+\frac{1}{\Delta}).$$

Two distinct job lengths				
	$D_2(\omega,\Delta)$		$R_2(\omega,\Delta)$	
	LB	UB	LB	UB
$0 \le \omega < \frac{1}{\Delta}$	$\max(1+\Delta, 2+\frac{1}{\Delta})$ (Cor. 24)	(Thm. 5)	$2-\frac{1}{\Delta}$ [8]	4 [6]
$\frac{1}{\Delta} \le \omega < 1$	$2+\frac{\lceil\Delta\rceil-1}{\Delta}$ (Thm. 14)	(Thm. 6)		
$1 \le \omega < \Delta$	$1+\frac{\lceil\Delta\rceil}{\Delta}$ (Thm. 15)	(Thm. 8)		
$\omega \ge \Delta$	$1+\frac{\lceil\Delta\rceil}{\Delta+\lfloor\omega-\Delta\rfloor+1}$ (Thm. 25)	$2+\frac{1}{\lceil\omega\rceil}$ (Thm. 11)		

Arbitrary job lengths				
	$D(\omega,\Delta)$		$R(\omega,\Delta)$	
	LB	UB	LB	UB
$0 \le \omega < \frac{1}{\Delta}$	$2+\Delta$ (Thm. 18)	(Thm. 23)	$\Omega(\log\Delta)$ [8]	$6(\lceil\lg_2\Delta\rceil+1)$ [6]
$\frac{1}{\Delta} \le \omega \le \lceil\Delta\rceil-1$	$2+\frac{1}{\omega}$ (Thms. 19,20)	(Thm. 10)		
$\lceil\Delta\rceil-1 < \omega < \Delta$	$2+\frac{1}{\omega}-\frac{\{\omega\}}{\omega}$ (Cor. 26)	$2+\frac{1}{\omega}$ (Thm. 10)		
$\omega \ge \Delta$ $[0<\{\Delta\}\le\{\omega\}]$	$1+\frac{\lceil\Delta\rceil}{\lfloor\omega\rfloor+1}$ (Cor. 26)	$2+\frac{1}{\lceil\omega\rceil}$ (Thm. 11)		
$\omega \ge \Delta$ $[\{\Delta\}=0$ or $\{\Delta\}>\{\omega\}]$	$1+\frac{\lceil\Delta\rceil}{\omega}$ (Cor. 26)	$2+\frac{1}{\lceil\omega\rceil}$ (Thm. 11)		

Table 2: The lower/upper bounds for the deterministic/randomized competitiveness of the problem, when parameterized both by ω and Δ.

Proof: This proof is the direct result of applying Theorem 14 when $\Delta < 2$, combined with the bound from Theorem 13. ∎

Theorem 25 *If* $\omega \geq \Delta - 1$,

$$D_2(\omega, \Delta) \geq 1 + \frac{\lceil \Delta \rceil}{\Delta + \lfloor \omega - \Delta \rfloor + 1}.$$

Proof: For this proof, we let $\{\Delta\} = \Delta - \lfloor \Delta \rfloor$ denote the fractional part of Δ, and $\{\omega\} = \omega - \lfloor \omega \rfloor$ the fractional part of ω. We break our proof into three separate cases.

The first case is if $\{\Delta\} = 0$ and the second case is if $\{\omega\} < \{\Delta\}$. We use the identical construction for both of these cases, with only a slight difference in the analysis. We let $\epsilon > 0$ be an arbitrarily small constant, and we consider the behavior of an algorithm A when faced with job $J_1 = \langle 0, \Delta, \Delta + \omega + 2 \rangle$. If A is deterministic, assume it begins running job J_1 at time t. Now we consider an additional $\lfloor \omega \rfloor + 1$ small jobs, each with parameters $\langle \epsilon, 1, \omega \rangle$. The expiration time for all new jobs is equal to $t + \omega + \epsilon$. Since A runs job J_1 from $[t, t + \Delta)$, the only small jobs it can run must be started during the interval $[t + \Delta, t + \omega + \epsilon]$. But at most $\max(0, 1 + \lfloor (t + \omega + \epsilon) - (t + \Delta) \rfloor) = \max(0, 1 + \lfloor \omega + \epsilon - \Delta \rfloor)$ small jobs can be started during that interval. For sufficiently small ϵ, $1 + \lfloor \omega + \epsilon - \Delta \rfloor = 1 + \lfloor \omega - \Delta \rfloor$. Since $\omega \geq \Delta - 1$, $\max(0, 1 + \lfloor \omega - \Delta \rfloor) = 1 + \lfloor \omega - \Delta \rfloor$. So the gain of A including J_1 is at most $\Delta + \lfloor \omega - \Delta \rfloor + 1$. The optimal schedule achieves all job for a gain of $\Delta + \lfloor \omega \rfloor + 1$, by running all of the small jobs from $[t, t + \lfloor \omega \rfloor + 1)$, and running the large job either before or after. Therefore, the competitiveness of any such algorithm A is no better than,

$$\frac{\Delta + \lfloor \omega \rfloor + 1}{\Delta + \lfloor \omega - \Delta \rfloor + 1} = 1 + \frac{\lfloor \omega \rfloor - \lfloor \omega - \Delta \rfloor}{\Delta + \lfloor \omega - \Delta \rfloor + 1}.$$

In both cases, we are able to show that the numerator is exactly equal to $\lceil \Delta \rceil$, proving our claim. In the case where $\{\Delta\} = 0$ and thus $\Delta = \lceil \Delta \rceil$, we see that,

$$\lfloor \omega \rfloor - \lfloor \omega - \Delta \rfloor = \lfloor \omega \rfloor - \lfloor \omega - \lceil \Delta \rceil \rfloor = \lfloor \omega \rfloor - (\lfloor \omega \rfloor - \lceil \Delta \rceil) = \lceil \Delta \rceil.$$

In the second case, where $\{\omega\} < \{\Delta\}$, we see that

$$\begin{aligned} \lfloor \omega \rfloor - \lfloor \omega - \Delta \rfloor &= \lfloor \omega \rfloor + \lceil \Delta - \omega \rceil = \lfloor \omega \rfloor + \lceil \Delta - \lfloor \omega \rfloor - \{\omega\} \rceil = \lfloor \omega \rfloor - \lfloor \omega \rfloor + \lceil \Delta - \{\omega\} \rceil \\ &= \lceil \Delta - \{\omega\} \rceil = \lceil \lfloor \Delta \rfloor + \{\Delta\} - \{\omega\} \rceil = \lfloor \Delta \rfloor + \lceil \{\Delta\} - \{\omega\} \rceil = \lfloor \Delta \rfloor + 1 \\ &= \lceil \Delta \rceil. \end{aligned}$$

Finally, the remaining case is if $0 < \{\Delta\} \leq \{\omega\}$. In this case we again consider the behavior of an algorithm A when faced with job $J_1 = \langle 0, \Delta, \Delta + \omega + 2 \rangle$. If A is deterministic, assume it begins running job J_1 at time t. This time, we change the construction slightly, considering an additional $\lceil \omega \rceil + 1$ small jobs, each with parameters $\langle \epsilon, 1, \lceil \omega \rceil \rangle$. The expiration time for all new jobs is equal to $t + \lceil \omega \rceil + \epsilon$. Since A runs job J_1 from $[t, t + \Delta)$, the only small jobs it can run must be started during the interval $[t + \Delta, t + \lceil \omega \rceil + \epsilon]$. During this

interval, at most $\max(0, 1 + \lfloor (t + \lceil \omega \rceil + \epsilon) - (t + \Delta) \rfloor) = \max(0, 1 + \lfloor \lceil \omega \rceil + \epsilon - \Delta \rfloor)$ small jobs can be started. We see that,

$$\begin{aligned} 1 + \lfloor \lceil \omega \rceil + \epsilon - \Delta \rfloor &= 1 + \lceil \omega \rceil + \lfloor \epsilon - \Delta \rfloor = 1 + \lceil \omega \rceil - \lceil \Delta - \epsilon \rceil = 1 + \lceil \omega \rceil - \lceil \Delta \rceil \\ &= 1 + \lfloor \omega \rfloor - \lfloor \Delta \rfloor = 1 + \lfloor \omega \rfloor - \lfloor \Delta \rfloor + \lfloor \{\omega\} - \{\Delta\} \rfloor \\ &= 1 + \lfloor \lfloor \omega \rfloor + \{\omega\} - \lfloor \Delta \rfloor - \{\Delta\} \rfloor = 1 + \lfloor \omega - \Delta \rfloor. \end{aligned}$$

Since $\omega \geq \Delta - 1$, $\max(0, 1 + \lfloor \omega - \Delta \rfloor) = 1 + \lfloor \omega - \Delta \rfloor$. So the gain of A including J_1 is at most $\Delta + \lfloor \omega - \Delta \rfloor + 1$. The optimal schedule achieves all job for a gain of $\Delta + \lceil \omega \rceil + 1$, by running all of the small jobs from $[t, t + \lceil \omega \rceil + 1)$, and running the large job either before or after. Therefore, the competitiveness of any such algorithm A is no better than,

$$\frac{\Delta + \lceil \omega \rceil + 1}{\Delta + \lfloor \omega - \Delta \rfloor + 1} = 1 + \frac{\lceil \omega \rceil - \lfloor \omega - \Delta \rfloor}{\Delta + \lfloor \omega - \Delta \rfloor + 1}.$$

Rearranging the numerator proves our claims, as $\lceil \omega \rceil = \lfloor \omega \rfloor + 1$ and $\lceil \Delta \rceil = \lfloor \Delta \rfloor + 1$, and so

$$\begin{aligned} \lceil \omega \rceil - \lfloor \omega - \Delta \rfloor &= \lfloor \omega \rfloor + 1 - \lfloor \omega - \Delta \rfloor = \lfloor \omega \rfloor + 1 - \lfloor \lfloor \omega \rfloor + \{\omega\} - \Delta \rfloor \\ &= \lfloor \omega \rfloor - \lfloor \omega \rfloor + 1 - \lfloor \{\omega\} - \Delta \rfloor = 1 - \lfloor \{\omega\} - \Delta \rfloor = 1 - \lfloor \{\omega\} - \lfloor \Delta \rfloor - \{\Delta\} \rfloor \\ &= 1 - (\lfloor \{\omega\} - \{\Delta\} \rfloor - \lfloor \Delta \rfloor) = 1 + \lfloor \Delta \rfloor - \lfloor \{\omega\} - \{\Delta\} \rfloor = 1 + \lfloor \Delta \rfloor = \lceil \Delta \rceil. \end{aligned}$$

∎

Corollary 26 *For all values of* $\omega > \lceil \Delta \rceil - 1$,

$$\textit{when } 0 < \{\Delta\} \leq \{\omega\}, \qquad D(\omega, \Delta) \geq 1 + \frac{\lceil \Delta \rceil}{\lfloor \omega \rfloor + 1}.$$

$$\textit{when } \{\Delta\} = 0 \textit{ or } \{\Delta\} > \{\omega\}, \qquad D(\omega, \Delta) \geq 1 + \frac{\lceil \Delta \rceil}{\omega},$$

Proof: We know that $D(\omega, \Delta) \geq D(\omega, \Delta') \geq D_2(\omega, \Delta')$ so long as $\Delta' \leq \Delta$.

First we consider the case that $0 < \{\Delta\} \leq \{\omega\}$. We let ϵ be an arbitrarily small constant such that $0 < \epsilon < \max(\{\omega\}, \omega - (\lceil \Delta \rceil - 1))$ and let $\Delta' = \lceil \Delta \rceil - 1 + \epsilon$. It is easy to verify that $\omega > \Delta' - 1$, and so Theorem 25 assures us that,

$$\begin{aligned} D_2(\omega, \Delta') &\geq 1 + \frac{\lceil \Delta' \rceil}{\Delta' + \lfloor \omega - \Delta' \rfloor + 1} = 1 + \frac{\lceil \lceil \Delta \rceil - 1 + \epsilon \rceil}{\lceil \Delta \rceil - 1 + \epsilon + \lfloor \omega - (\lceil \Delta \rceil - 1 + \epsilon) \rfloor + 1} \\ &= 1 + \frac{\lceil \Delta \rceil - 1 + \lceil \epsilon \rceil}{\lceil \Delta \rceil - 1 + \epsilon - \lceil \Delta \rceil + 1 + \lfloor \omega - \epsilon \rfloor + 1} = 1 + \frac{\lceil \Delta \rceil}{\epsilon + \lfloor \omega \rfloor + 1} \end{aligned}$$

As $\Delta' \leq \Delta$, we know that $D(\omega, \Delta) \geq D_2(\omega, \Delta')$, and since ϵ can be chosen to be arbitrarily small, we get that $D(\omega, \Delta) \geq 1 + \frac{\lceil \Delta \rceil}{\lfloor \omega \rfloor + 1}$.

In the case that $\{\Delta\} > \{\omega\}$, let ϵ be an arbitrarily small constant such that $0 < \epsilon < \{\Delta\} - \{\omega\}$ and in the case where $\{\Delta\} = 0$, let ϵ be an arbitrarily small constant such that $0 < \epsilon < 1 - \{\omega\}$. Now let $\Delta' = \lceil \Delta \rceil - 1 + \{\omega\} + \epsilon$. It is easy to verify that $\omega > \Delta' - 1$, and so Theorem 25 assures us that,

$$\begin{aligned}
D_2(\omega, \Delta') &\geq 1 + \frac{\lceil \Delta' \rceil}{\Delta' + \lfloor \omega - \Delta' \rfloor + 1} \\
&= 1 + \frac{\lceil \lceil \Delta \rceil - 1 + \{\omega\} + \epsilon \rceil}{\lceil \Delta \rceil - 1 + \{\omega\} + \epsilon + \lfloor \omega - (\lceil \Delta \rceil - 1 + \{\omega\} + \epsilon) \rfloor + 1} \\
&= 1 + \frac{\lceil \Delta \rceil - 1 + \lceil \{\omega\} + \epsilon \rceil}{\lceil \Delta \rceil - 1 + \{\omega\} + \epsilon - \lceil \Delta \rceil + 1 + \lfloor \omega - \{\omega\} - \epsilon \rfloor + 1} \\
&= 1 + \frac{\lceil \Delta \rceil}{\{\omega\} + \epsilon + \lfloor \lfloor \omega \rfloor - \epsilon \rfloor + 1} = 1 + \frac{\lceil \Delta \rceil}{\lfloor \omega \rfloor + \{\omega\} + \epsilon + \lfloor -\epsilon \rfloor + 1} = 1 + \frac{\lceil \Delta \rceil}{\omega + \epsilon}
\end{aligned}$$

As $\Delta' \leq \Delta$, we know that $D(\omega, \Delta) \geq D_2(\omega, \Delta')$, and since ϵ can be chosen to be arbitrarily small, we get that $D(\omega, \Delta) \geq 1 + \frac{\lceil \Delta \rceil}{\omega}$. ∎