Proc. of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99), January 1999, pp. 396–405.

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

Michael H. Goldwasser*

Abstract

We consider the online competitiveness for scheduling a single resource non-preemptively in order to maximize its utilization. The *slack* of a job is equal to the gap between its release time and the last possible time at which it may be started while still meeting its deadline. When no restrictions are placed on either the length or slack of any job, previous lower bounds show that no algorithm, deterministic or randomized, can guarantee any constant bound on the competitiveness of a resulting schedule. We examine a natural restriction in which each job must offer slack at least proportional to the job's processing time. Specifically, we will say that a problem instance has *patience* κ, if any job of length $\|J\|$ has a slack of at least $\kappa \cdot \|J\|$. We show that for any $\kappa > 0$ a simple greedy algorithm is $(2+\frac{1}{\kappa})$-competitive even when arbitrary job lengths are allowed. We give lower bounds showing that this is the best possible result for a deterministic algorithm even if all jobs have one of three distinct lengths. In the special case where all jobs have the same length, we generalize a previous bound of 2 for the deterministic competitiveness with arbitrary slacks, showing that the competitiveness for any $\kappa \geq 0$ is exactly $1+\frac{1}{\lfloor \kappa \rfloor + 1}$. We also give tight bounds for the case where jobs have one of two distinct lengths.

1 Introduction

Imagine that a company rents use of a single resource, charging customers a fixed amount per minute. Job requests arrive, with each request specifying the length of time for which the customer will use the resource as well as a deadline by which the job should be completed. In the terminology of real-time systems we consider this a *firm deadline* [14]. The company is not required to complete each job, however it will only be paid for those jobs that are completed on or before their deadline. Therefore, a scheduling algorithm provides admission control as it decides which jobs to run, as well as when to run them. The goal is to maximize the successful use of the resource and thus the profit for the company. We require that a schedule be *non-preemptive*, in that once a customer's job begins running, the job cannot be interrupted or terminated. The problem is online in nature as we assume that an algorithm has no knowledge about the existence of any job until the time at which a request arrives. Once a job does arrive, the algorithm is immediately told both the length and deadline of the job. We analyze the performance of an algorithm using *competitive analysis*, comparing the schedule produced by an algorithm to the optimal schedule produced by an algorithm with full knowledge of future jobs [13, 20].

This model arises in many applications of real-time systems, with the most typical motivations involving communication-based applications. In this setting, the resource represents a channel of communication and a job (or call) represents a request for delivery of text, sound, or video through the channel. For example, a video-on-demand provider has 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 with some deadline for completion. In these applications, the insistence on non-preemptive scheduling is due to customer relations. A provider may simply fail to satisfy a request, however it may not begin servicing a request only to terminate it because a better customer comes along.

The *slack*[1] of a job is equal to the amount of time between the job's arrival and the last possible time at which it could be started while still meeting its deadline. For example, the request of an impatient customer may have zero slack, meaning that this job must be given the resource immediately or else rejected. Our paper focuses on the effect of slack on the competitiveness of scheduling algorithms in this model. Specifically we say that a problem instance has *patience* κ, if a job J of length $\|J\|$ offers a slack of at least $\kappa\|J\|$. For example, if the patience of an instance is 0.05 then a customer requesting 5 minutes of a resource must be willing to wait at least 15 seconds before beginning, whereas a

*Dept. of Computer Science, Princeton University, Princeton, NJ 08544. Email: wass@cs.princeton.edu. Research at Princeton University supported by DIMACS (Center for Discrete Mathematics and Theoretical Computer Science), a National Science Foundation Science and Technology Center.

[1] This property has also been referred to as *delay*, *wait time* or *laxity* by previous researchers.

customer requesting 5 hours of time must be willing to wait at least 15 minutes instead. Measuring the slack in relation to a job's processing time is quite natural for admission control and similar restrictions have been examined in related models [4, 6, 8, 10].

The main result of our paper is that the competitiveness of online scheduling algorithms is significantly improved when instances have non-zero patience. When arbitrary slacks are allowed (i.e., when $\kappa = 0$), a previous lower bound shows that even if randomization is allowed, the competitiveness depends logarithmically on Δ, the ratio between the maximum and minimum job lengths [16]. Therefore, if arbitrary job lengths and slacks are allowed, no bounded competitiveness is possible. However, for all values of $\kappa > 0$, we show that a simple deterministic algorithm is $(2 + \frac{1}{\kappa})$-competitive, thereby providing a constant competitive ratio that does not depend on the range of job lengths. We provide a matching 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), we show that the natural greedy algorithm is $(1 + \frac{1}{\lfloor \kappa \rfloor + 1})$-competitive, and again that this is the best possible deterministic result. This result generalizes a previous bound of 2-competiveness for the case when $\kappa = 0$ [10]. In the case where jobs have one of two distinct lengths, we give tight upper and lower bounds which improve slightly on the $(2 + \frac{1}{\kappa})$ bound for arbitrary lengths.

A complete summary of our upper and lower bounds when parameterized by κ is given in Table 1. A more complete set of bounds when parameterized by both κ and Δ is given in the appendix. Our results are robust in the following manner. All of our lower bounds apply even when the values of both κ and Δ are known to the algorithm. At the same time, all of our upper bounds are achieved by algorithms that have no knowledge about the values of κ or Δ. Furthermore, our lower bounds assume that an individual job's slack time and processing length are given to an algorithm when the job request arrives. However, our general algorithm for the case of arbitrary job lengths needs no advanced knowledge of either of these properties[2].

2 Notation

We consider a job J_j to be a triple of non-negative integers $\langle r_j, p_j, s_j \rangle$, where r_j is the release time or arrival time of the job, p_j is the length of the processing time, and s_j is the slack. We define the patience of an instance as $\kappa = \min_j(s_j/p_j)$, so that every job J_j with processing time p_j has a slack $s_j \geq \kappa \cdot p_j$. We will refer to the *deadline* of the job, $d_j = r_j + s_j + p_j$, as the time at which the job must be completely processed, and the *expiration time*, $x_j = r_j + s_j$, as the latest time at which a job can be started while still meeting its deadline. We say that a job J_j is *available* at time t with respect to a schedule σ if $r_j \leq t \leq x_j$ and if job J_j has not been started in σ prior to time t. We will also let $\|J_j\|$ denote the processing time p_j. The *gain* of a schedule σ, which we denote as $\|\sigma\|$, is equal to $\sum_{J \in \sigma} \|J\|$ and our goal is to maximize the gain. Following the standard notation of Graham *et. al.* [11, 15], our general problem is exactly an online version of $1 \mid r_j \mid \sum p_j \overline{U}_j$.

2.1 Competitive Analysis. We will measure the performance of an online algorithm by comparing the gain of the algorithm to the gain of the optimal offline algorithm that knows the entire future when making decisions [13, 20]. We will say that a (deterministic) online algorithm A is *c-competitive*[3] if $\text{gain}_{opt}(\mathcal{I}) \leq c \cdot \text{gain}_A(\mathcal{I})$, for all input instances $\mathcal{I}$. Finally, we define the competitiveness of the problem, $\mathcal{C}$, 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 performance 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 a worst case input for an algorithm is chosen by an *oblivious* adversary, which must choose the entire sequence in advance [7, 19].

In order to characterize the competitiveness of this particular scheduling problem based on different values of κ, we introduce the following notation. We let $\hat{D}(\kappa)$ denote the competitiveness of the problem for a fixed κ without any restrictions on the job lengths. 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 $\hat{D}_1(\kappa)$ to be the deterministic competitiveness for the case where all jobs must have equal length, and we define $\hat{D}_2(\kappa)$ to be the competitiveness for the case where all jobs must have one of two distinct lengths. In general, we see that

[2] The algorithm for equal length jobs utilizes the slack times in its decision making, and the algorithm for two distinct lengths utilizes the slack times and job lengths in its decision making.

[3] 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.

Equal length jobs				
	$\hat{D}_1(\kappa)$		$\hat{R}_1(\kappa)$	
	LB	UB	LB	UB
$\kappa = 0$	2 From [10]		4/3 From [10]	
$\kappa \geq 0$	$1 + \frac{1}{\lfloor \kappa \rfloor + 1}$		$1 + \frac{1}{2\lfloor \kappa \rfloor + 3}$	

Two distinct job lengths				
	$\hat{D}_2(\kappa)$		$\hat{R}_2(\kappa)$	
	LB	UB	LB	UB
$\kappa = 0$	∞		2 From [16]	4 From [10]
$\kappa > 0$	$1 + \max(\frac{\lceil \kappa \rceil + 1}{\lceil \kappa \rceil}, \frac{\lfloor \kappa \rfloor + 1}{\kappa})$			

Arbitrary job lengths				
	$\hat{D}(\kappa)$		$\hat{R}(\kappa)$	
	LB	UB	LB	UB
$\kappa = 0$	∞		∞ From [16]	
$\kappa > 0$	$2 + \frac{1}{\kappa}$			

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

$\hat{D}(\kappa) \geq \hat{D}_2(\kappa) \geq \hat{D}_1(\kappa)$, as each expression from left to right refers to a more restricted set of problem instances. By replacing the letter "D" with the letter "R" in our notation, we denote the *randomized* competitiveness of the respective problems. Since any deterministic algorithm can be viewed as a randomized algorithm, we immediately have that $\hat{R}_i(\kappa) \leq \hat{D}_i(\kappa)$ in general.

3 Previous Work

The problem of online, non-preemptive scheduling of a single resource to maximize resource utilization was studied by Lipton and Tomkins in the case when all jobs have zero slack [16]. This problem, termed *online interval scheduling*, can be solved optimally when all jobs have equal length. In the case where all jobs have lengths p or $p \cdot \Delta$, they provide a randomized online algorithm that is 2-competitive for any Δ. A lower bound of $2 - \frac{1}{\Delta}$ shows that the upper bound is the best possible as Δ increases. When arbitrary job lengths are considered, they provide a lower bound of $\Omega(\log \Delta)$ for the competitiveness of any randomized algorithm. 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 [17].

Goldman, Parwatikar and Suri extended the model of Lipton and Tomkins, allowing jobs to specify arbitrary slacks [10]. They make no assumptions on the slacks specified by jobs and so this setting corresponds to the case $\kappa = 0$ in our model. The existence of slack acts as a double-edged sword for competitive analysis. Clearly the added flexibility can only serve to increase the resource utilization for an algorithm. However maintaining competitiveness may become more difficult, as the optimal offline algorithm may know how to take better advantage of the flexibility than an online algorithm. 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, showing that a deterministic greedy algorithm is 2-competitive and that this is the best possible result. They give a lower bound of $\frac{4}{3}$ for the randomized competitiveness in this setting, however without a matching upper bound. Additionally they prove a curious result, namely that if all slacks are equal to 1 or greater, then the same greedy algorithm becomes $\frac{3}{2}$-competitive. It is this result that is really the springboard of our own work. Our work generalizes this result for all values of κ and re-examines the cases where job lengths vary. Goldman *et. al.* also consider a model in which they require that all jobs of the same length have the same slack, although the value of this slack need not have any relationship to the actual job length. With this additional requirement which they call *uniform delays*, 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. Although this model of uniform delays does not readily translate to our model, as κ increases our results are significantly stronger than the bounds given for uniform delays.

Previous researchers have studied several closely related problems. Baruah *et. al.* study the *preemptive* setting of our problem, giving a 4-competitive algorithm for maximizing the use of a single processor with arbitrary job lengths and slacks [5]. They also show that this is the best possible (deterministic) algorithm. Scheduling a single resource is a special case of the more general problem of call control in larger communication networks, where both admission and routing are issues. The competitiveness of various call control models has been studied in both the non-preemptive [1, 2] and preemptive [3, 9] settings, with a more complete survey given by Plotkin [18].

Finally, the effect of requiring slack times proportional to processing times has been studied in several other models. The advantage of patience on competitiveness for scheduling single processors *preemptively* has been studied. Baruah and Haritsa give almost tight bounds for maximizing the utilization of a single processor [6], and Kalyanasundaram and Pruhs consider more general benefits associated with each job's completion [12]. The advantage of increased slack on competitiveness is also shown by Bar-Noy *et. al.* for a different problem specifically modeling movies-on-demand [4]. Their model assumes equal job lengths, as well as equal slack times, however in a setting where multiple requests can be serviced by a single channel if the requests are for the same movie. Finally, in a non-preemptive setting, Feldman *et. al.* study the requirement that slack time be proportional to processing time in the context of call control on networks [8]. In contrast to our results, they show that this additional requirement cannot be used to asymptotically improve the competitiveness of algorithms on networks of size n. They generalize previous lower bounds to include proportional slack, even when a network is a linear array and randomization is allowed. In light of this, our results demonstrate that this lack of improvement is inherently due to the routing aspects of the general problem and not due to the non-preemptive admission control.

4 Equal Length Jobs

For this section, we concern ourselves only with equal length jobs. Without loss of generality, we scale units of time so that each job has length 1, allowing non-integral release times and deadlines if necessary. We begin by proving a lower bound parameterized by κ for both the deterministic and randomized competitiveness. Following this, we give a tight upper bound for the deterministic competitiveness for all values of κ.

4.1 Lower Bounds.

THEOREM 4.1. *For all κ,*

$$\hat{D}_1(\kappa) \geq 1 + \frac{1}{\lfloor \kappa \rfloor + 1} \qquad \textit{and} \qquad \hat{R}_1(\kappa) \geq 1 + \frac{1}{2\lfloor \kappa \rfloor + 3}.$$

Proof: Let $z = \lfloor \kappa \rfloor + 1$, and thus $\kappa = z - 2\epsilon$ for some $0 < \epsilon \leq \frac{1}{2}$. Consider the following two scenarios, consisting of $z + 1$ jobs with minimum slack κ. 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 z other 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.

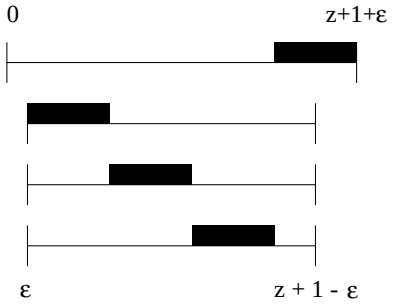

Figure 1: Arrivals and deadlines for instance $\mathcal{S}_1$, in which it is necessary to schedule job J_1 last.

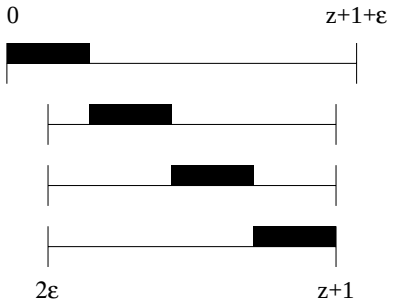

Figure 2: Arrivals and deadlines for instance $\mathcal{S}_2$, in which it is necessary to schedule job J_1 at time $t = 0$.

Our second scenario, $\mathcal{S}_2$, consists of the same first job J_1, with 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 other z jobs from time $[1, z + 1)$. Again, any other schedule runs at most z of the jobs. These scenarios are shown in Figures 1-2.

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 $t = 0$. For either behavior, one of the two scenarios above will result in only z tasks being scheduled, although the optimal algorithm achieves all $z + 1$ tasks. Therefore A's competitive ratio is at least $\frac{z+1}{z} = 1 + \frac{1}{z} = 1 + \frac{1}{\lfloor\kappa\rfloor+1}$.

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 $\frac{z+1}{z+\frac{1}{2}} = 1 + \frac{1}{2z+1} = 1 + \frac{1}{2\lfloor\kappa\rfloor+3}$. □

`Filter`(σ, σ^*)
(1) Initially set σ' equal to σ.
(2) Scan through σ', from beginning to end
(3) At time t when σ' starts a job $J \notin \sigma^*$
(or a MARKER from line 9 exists),
(4) Let set $\mathcal{A}$ consist of those jobs of σ^*
which are available to σ' at time t
(5) If $\mathcal{A}$ is non-empty
(6) Pick $J_a \in \mathcal{A}$ with earliest expiration
(7) Remove J from σ' and replace it with J_a
(8) If J_a is scheduled at a later time in σ',
(9) Remove J_a from that later spot of σ'
(and replace with a MARKER).
(10) Remove any MARKER's, leaving σ' idle.

Figure 3: Creating an intermediary schedlue σ'.

4.2 Deterministic Upper Bound. 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 $1 + \frac{1}{\lfloor\kappa\rfloor+1}$.

As in [10], 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 release 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. Therefore, 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.

Rather than analyze the true schedule σ, we construct an intermediary schedule σ' and base our analysis on it. This intermediary schedule is identical to σ, except that when possible we complete jobs common to σ^* in lieu of jobs that are only run by σ. Given both σ and σ^*, we construct σ' as shown in Figure 3. The rest of our proof will rely on the following two technical lemmas.

LEMMA 4.1. *For any time t at which σ originally started a job, if $J_k \in \sigma^*$ is available to σ' at that time, then σ' must start some job $J_i \in \sigma^*$ with $x_i \leq x_k$ at time t.*

Proof: Let t be some original start time of σ and let J_k be a job of σ^* which was available to σ' at that time. If this time slot of σ' contained a job not in σ^* or a MARKER, our construction surely would have instead placed J_k or some other job of σ^* in this slot at line 7 of `Filter`. So some job $J_i \in \sigma^*$ must be started in σ' at this time.

J_i's placement in σ' could be for one of two reasons. If J_i was originally scheduled in this slot in 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 certainly J_k was available. □

LEMMA 4.2. *If σ' starts some job at a given time, there can not exist 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 r is the earlier release time of the two jobs, 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 $t \leq x < t+1$). Notice that Lemma 4.1 says that the job scheduled at time t must be in σ^* with an expiration time less than or equal to x. Continuing in this manner, we define s to be the earliest start time in σ' such that all jobs that σ' starts over the range $[s,t]$ are jobs of σ^* with expiration times of x or earlier. Since the intervals of availability for our two jobs overlap, at least one of them was available to σ' throughout the range $[r,t]$. Therefore, Lemma 4.1 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 $s \leq \lceil r \rceil \leq r+1$. A sample range $[s,t]$ is displayed in Figure 4.

Let us denote as B, the set of all jobs that σ' began during $[s,t]$, together with jobs J_1 and J_2. Notice that $|B| = 2 + (t+1-s) = t-s+3$. We have already seen that every job of B has an expiration $x_i \leq x < t+1$ and belongs to σ^*. We also claim that each of these jobs has a release time $r_i > s-1$. To see this, notice that if a job in B had arrived on or before $s-1$, it was available to σ' at time $s-1$. Lemma 4.1 applies in this case, and so the job started at time $s-1$ must be a member of σ^* with an expiration time of x or less. However, this contradicts our choice of s, and so we conclude that each

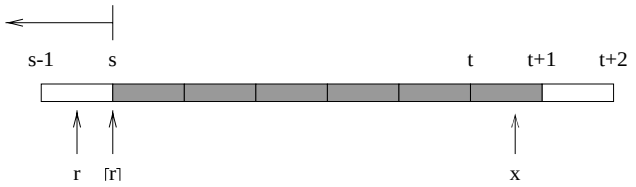

Figure 4: The range $[s,t]$ in σ', with jobs in B shaded.

member of set B was indeed released strictly after time $s-1$.

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 B. This is a contradiction, as all members of B are contained in σ^*. □

Now, we complete our analysis of GREEDY. We have assumed that the GREEDY schedule σ never idles, and thus runs throughout the interval $[0,e)$ for some e. We let $\mathcal{L}$ denote the jobs in σ^* scheduled to begin on or after time e (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 $\mathcal{L}$ (i.e., 'non-late'). 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 any such late job must already be completed in σ. Therefore σ is exactly $\mathcal{N} \cup \mathcal{L}$ and so $\|\sigma\| = N+L$. Finally, 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.

It is known that GREEDY is 2-competitive when $\kappa = 0$ [10]. This bound can be seen from the above argument as $P \leq N+L = \sigma$ and $L \leq \sigma$, and so $\sigma^* = P+L \leq 2\sigma$. In order to improve the analysis for larger values of κ, we will show a lower bound on the number of jobs in set $\mathcal{N}$.

If we re-examine our construction of σ', we see that $\|\sigma'\| \leq \|\sigma\|$. If we analogously define $\mathcal{N}'$, we again see that $\|\sigma'\| = N'+L$ as our construction of σ' would never have thrown out any elements of $\mathcal{L} \subseteq \sigma^*$. Therefore $\|\sigma'\| \leq \|\sigma\|$ implies that $N' \leq N$.

LEMMA 4.3.

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

Proof: Since $N \geq N'$, we devise a charging scheme where each job of $\mathcal{N}'$ is allowed to distribute up to $\lfloor \kappa \rfloor + 1$

units of charge, and each job of $\mathcal{P}$ receives at least $\lfloor\kappa\rfloor$ units. This scheme shows that $(\lfloor\kappa\rfloor+1)N' \geq \lfloor\kappa\rfloor P$, proving our claim.

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

- $\lfloor\kappa\rfloor$ is assigned to itself, if $J_i \in \sigma^*$.
- 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 $\lfloor\kappa\rfloor+1$ units overall. This is a result of Lemma 4.2, 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 $\lfloor\kappa\rfloor$. If a job $J_k \in \mathcal{P}$ was run in σ', then that job will have been assigned $\lfloor\kappa\rfloor$ from the first rule, and we are done. We must only show this result for a job $J_k \in \mathcal{P}$ that did not run in σ'. First we note that such a job must not be scheduled in σ either (our construction of σ' would never have thrown out a member of $\mathcal{P} \subseteq \sigma^*$), and thus it must be that $x_k < e$. Since this job has a minimum slack of at least $\kappa \cdot \|J\|$, there must have been at least $\lfloor\kappa\rfloor$ original start times at which this job was available to σ'. By Lemma 4.1, 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 start times at which J_k was available to σ', it must be that J_k has been assigned at least $\lfloor\kappa\rfloor$. □

THEOREM 4.2. *The* GREEDY *algorithm is strongly* $\left(1+\frac{1}{\lfloor\kappa\rfloor+1}\right)$*-competitive.*

Proof: This follows from Lemma 4.3, and the fact that $P \leq N + L$.

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

□

5 Two Distinct Job Lengths

In this section, we consider the case where all jobs have length either p or $p \cdot \Delta$ for some constant p and $\Delta > 1$. We will refer to jobs of length p as "small" jobs, and jobs of length $p\Delta$ as "large" jobs.

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 5.1. *For all* $\kappa > 0$, GREEDY-TWOLENGTHS *is* $\left(1+\max(\frac{\lceil\kappa\rceil+1}{\lceil\kappa\rceil}, \frac{\lfloor\kappa\rfloor+1}{\kappa})\right)$*-competitive, and this is the best possible result for a deterministic algorithm.*

The proof is given in the full version.

6 Arbitrary Job Lenghts

In this section, we consider the case where jobs can have arbitrary lengths, and where the minimum allowable slack for any job of length $\|J\|$ is equal to $\kappa\|J\|$. It is worth noting that our upper bound is proven with no apriori knowledge of the ratio between the smallest and largest job lengths, and with no knowledge of the patience value κ. In fact the algorithm works even if the processing time of a started job is not known until the job completes. In contrast, our matching lower bound is proven even if a job's length is known at arrival time, and if all jobs have one of three distinct lengths which are known at the onset of the algorithm.

THEOREM 6.1. *For* $\kappa > 0$,

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

Proof: Let ϵ be an arbitrarily small constant such that $e > 0$. We consider the behavior of a deterministic algorithm A when faced with job $J_1 = \langle 0, \lceil\kappa\rceil + 2\epsilon, \left(2+\frac{1}{\kappa}\right)\lceil\kappa\rceil + 3\epsilon\rangle$. As this job may be the only one to arrive, there must exist some time t at which A begins running job J_1, or else its competitiveness would be infinite. Now we consider an additional job $J_2 = \langle t+\epsilon, \frac{\lceil\kappa\rceil}{\kappa}, \lceil\kappa\rceil\rangle$, as well as $\lceil\kappa\rceil$ additional unit-length jobs, $\langle t+\epsilon, 1, \lceil\kappa\rceil\rangle$. Notice that all jobs satisfy the minimum slack requirement.

The expiration time for all additional jobs is equal to $t + \lceil\kappa\rceil + \epsilon$, and so A will miss them all, as it runs J_1 from $[t, t+\lceil\kappa\rceil+2\epsilon)$. Therefore, the gain of A is at most $\lceil\kappa\rceil + 2\epsilon$, whereas the optimal schedule can achieve all jobs for a gain of $\lceil\kappa\rceil\left(2+\frac{1}{\kappa}\right)+2\epsilon$, by running the small jobs from $[t+\epsilon, t+\lceil\kappa\rceil+\epsilon)$, and J_2 from

$[t+\lceil\kappa\rceil+\epsilon, t+\lceil\kappa\rceil\left(1+\frac{1}{\kappa}\right)+\epsilon)$. 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 $\frac{\lceil\kappa\rceil\left(2+\frac{1}{\kappa}\right)+2\epsilon}{\lceil\kappa\rceil+2\epsilon}$. Since ϵ can be chosen to be arbitrarily small, this lower bound can be made arbitrarily close to $2+\frac{1}{\kappa}$. □

THEOREM 6.2. *For $\kappa > 0$,*

$$\hat{D}(\kappa) \leq 2 + \frac{1}{\kappa}.$$

Proof: To prove this upper bound, 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. Although some rules may be better than others, it turns out that any such greedy-type algorithm is strongly competitive. Our proof structure is similar to the analysis of GREEDY in Section 4.2. Without loss of generality, we again assume that σ runs over the interval $[0, e)$ without idling.

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 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 exists, 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 $\|\sigma^*\| = P + \|J\| + L$, and again we have that $\|\sigma\| = N + L$, as we know that $\mathcal{L} \subseteq \sigma$.

If $J \in \mathcal{N}$ or if J does not exist, we see that $\|\sigma^*\| \leq 2\|\sigma\|$, as $\|J\| + L \leq N + L = \|\sigma\|$ and $P \leq \|\sigma\|$, and thus $\|\sigma^*\| = P + \|J\| + L \leq 2\|\sigma\|$. 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 $\kappa\|J\|$. 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 $\|\sigma\| \geq \kappa\|J\|$, that is $\|J\| \leq \frac{1}{\kappa}\|\sigma\|$. Combining this with the bounds $L \leq \|\sigma\|$ and $P \leq \|\sigma\|$, we see that $\|\sigma^*\| = P + \|J\| + L \leq \left(2 + \frac{1}{\kappa}\right)\|\sigma\|$. □

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 offer minimum slacks proportional to their jobs lengths. We show the following results,

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

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 [10] concerning the case of equal length jobs with no minimum slack. 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 $\kappa > 0$. Additionally, some gaps remain in the appendix, regarding the exact deterministic competitiveness when parameterized by both κ 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. We can also consider the weighted case in which each job specifies an additional parameter w_j signifying 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). Providing tight bounds in these settings based on patience remains open.

Acknowledgments. The author would like to thank Sally Goldman and Subhash Suri for many insights into their previous work on this problem, as well as Eric Torng and Sanjeev Arora for several helpful conversations. The author is additionally grateful to Sally for her detailed comments on an earlier draft of this paper.

References

[1] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive online routing. In *Proceedings of the 33rd Symposium on Foundations of Computer Science*, pages 32–40, 1993.

[2] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosén. Competitive non-preemptive call control. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 312–320, 1994.

[3] A. Bar-Noy, R. Canetti, S. Kutten, Y. Mansour, and B. Schiber. Bandwidth allocation with preemption. In *Proceedings of the 27th ACM Symposium on Theory of Computing*, pages 616–625, 1995.

[4] A. Bar-Noy, J. A. Garay, A. Herzberg, and S. Aggarwal. Sharing video on demand. Manuscript, 1998. Presented at the *Workshop on Algorithmic Aspects of Communication*, Bologna, Italy, July 1997.

[5] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasta, and F. Wang. On the competitiveness of on-line real-time task scheduling. In *Proceedings of the Real-Time Systems Symposium*, pages 106–115, 1991.

[6] S. K. Baruah and J. R. Harista. Scheduling for overload in real-time systems. *IEEE Transactions on Computers*, 46(9):1034–1039, 1997.

[7] 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.

[8] 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.

[9] J. Garey, I. Gopal, and S. Kutten. Efficient online call control algorithms. *Journal of Algorithms*, 23(1):180–194, 1997.

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

[11] R. Graham, E. L. Lawler, J. K. Lenstra, and A. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. In *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287–326. North Holland, 1979.

[12] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. In *Proceedings of the 36th Symposium on Foundations of Computer Science*, pages 214–221, 1995.

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

[14] H. Kopetz. *Real-time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Boston, 1997.

[15] E. L. Lawler, J. K. Lenstra, A. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S. Graves, A. Rinnooy Kan, and P. Zipken, editors, *Logistics of Production and Inventory*, volume 4 of *Handbooks in Operations Research and Management Science*, pages 445–522. North Holland, 1993.

[16] R. Lipton and A. Tomkins. Online interval scheduling. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 302–311, 1994.

[17] D. Long and M. Thakur. Scheduling realtime disk transfers for continuous media applications. In *Proceedings of the 12th IEEE Symposium on Mass Storage Systems*, pages 227–232, 1993.

[18] S. A. Plotkin. Competitive routing of virtual circuits in ATM networks. *IEEE Journal on Selected Areas in Communication*, 13(6):1128–1136, 1995.

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

[20] 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 (κ, Δ)

In the main paper, we gave tight bounds on the deterministic competitiveness of the problems, parameterized by the patience κ, irrespective of Δ, the ratio between the smallest and largest job lengths. For completeness, in this section we summarize our specific results for the competitiveness of the problem when both κ and Δ are fixed parameters.

In this regard, we define $D(\kappa, \Delta)$ to be the deterministic competitiveness for a fixed $\kappa \geq 0$ and $\Delta \geq 1$. We define $D_2(\kappa, \Delta)$ similarly for the case with two distinct job lengths. By our definitions, $\hat{D}(\kappa) = \max_\Delta D(\kappa, \Delta)$ and $\hat{D}_2(\kappa) = \max_\Delta D_2(\kappa, \Delta)$.

A complete summary of our upper and lower bounds when parameterized by both κ and Δ is given in Table 2.

Two distinct job lengths				
	$D_2(\kappa,\Delta)$		$R_2(\kappa,\Delta)$	
	LB	UB	LB	UB
$0 \le \kappa < \frac{1}{\Delta}$	$\max(1+\Delta, 2+\frac{1}{\Delta})$		$2-\frac{1}{\Delta}$ From [16]	4 From [10]
$\frac{1}{\Delta} \le \kappa < 1$	$2+\frac{\lceil\Delta\rceil-1}{\Delta}$			
$1 \le \kappa < \Delta$	$1+\frac{\lceil\Delta\rceil}{\Delta}$			
$\kappa \ge \Delta$	$1+\frac{\lceil\Delta\rceil}{\Delta+\lfloor\kappa-\Delta\rfloor+1}$	$2+\frac{1}{\lceil\kappa\rceil}$		

Arbitrary job lengths				
	$D(\kappa,\Delta)$		$R(\kappa,\Delta)$	
	LB	UB	LB	UB
$0 \le \kappa < \frac{1}{\Delta}$	$2+\Delta$		$\Omega(\log\Delta)$ From [16]	$6(\lceil\lg_2\Delta\rceil+1)$ From [10]
$\frac{1}{\Delta} \le \kappa \le \lceil\Delta\rceil-1$	$2+\frac{1}{\kappa}$			
$\lceil\Delta\rceil-1 < \kappa < \Delta$	$2+\frac{1}{\kappa}-\frac{\{\kappa\}}{\kappa}$	$2+\frac{1}{\kappa}$		
$\kappa \ge \Delta$ $[0<\{\Delta\}\le\{\kappa\}]$	$1+\frac{\lceil\Delta\rceil}{\lfloor\kappa\rfloor+1}$	$2+\frac{1}{\lceil\kappa\rceil}$		
$\kappa \ge \Delta$ $[\{\Delta\}=0$ or $\{\Delta\}>\{\kappa\}]$	$1+\frac{\lceil\Delta\rceil}{\kappa}$	$2+\frac{1}{\lceil\kappa\rceil}$		

Table 2: The lower/upper bounds for the deterministic/randomized competitiveness of the problem, when parameterized both by κ and Δ. The notation $\{x\}$ denotes the fractional part of a number, $\{x\} = x - \lfloor x \rfloor$.