Online story scheduling in web advertising

Anirban Dasgupta [*] Arpita Ghosh [*] Hamid Nazerzadeh [†‡]
Prabhakar Raghavan [*]

Abstract

We study an online job scheduling problem motivated by *storyboarding* in web advertising, where an advertiser derives value from having uninterrupted sequential access to a user surfing the web. The user ceases to browse with probability $1-\beta$ at each step, independently. *Stories* (jobs) arrive online; job s has a length ℓ_s and a per-unit value v_s. We get a value v_s for every unit of the job that we schedule consecutively without interruption, discounted for the time at which it is scheduled. Jobs can be preempted, with no further value derived from the residual unscheduled units of the job. We seek an online algorithm whose total reward is competitive against that of the offline scheduler that knows all jobs in advance.

We consider two models based on the maximum delay that can be allowed between the arrival and scheduling of a job. In the first, a job can be scheduled anytime after its arrival; in the second a job is lost unless scheduled immediately upon arrival, pre-empting a currently running job if needed. The two settings correspond to two natural models of how long an advertiser retains interest in a relevant user. We show that there is, in fact, a *sharp separation* between what an online scheduler can achieve in these two settings. In the first setting with no deadlines, we give a natural deterministic algorithm with a constant competitive ratio against the offline scheduler. In contrast, we show that in the sharp deadline setting, no (deterministic or randomized) online algorithm can achieve better than a polylogarithmic ratio.

[*]Yahoo! Research, Santa Clara, CA, email: {anirban, arpita, pragh}@yahoo-inc.com
[†]Stanford University, Stanford, CA, email: hamidnz@stanford.edu
[‡]Work performed in part at Yahoo! Research.

1 Introduction

Online advertising is a major source of revenue for Internet companies, with *display advertising* contributing a significant (21% [10]) and growing fraction. In display advertising, the content of a webpage – and increasingly, the browsing history of a user – is used for targeting ads.

One paradigm for targeting display ads, based on this ability to track a user across the web, is *storyboarding*, also referred to as sequence advertising or surround sessions [11, 8][1]. Here, a single advertiser gets exclusive access to a user for a sequence of consecutive pages viewed by the user, with no interruptions from other advertisers. This sequence of *slots* can be used by the advertiser either to show a set of unrelated ads for reinforcement of his message, or to creatively use a story line across several pages.

Consider a setting where multiple advertisers each vie for access to a sequence of contiguous steps during a user's browsing session. Each advertiser *appears* at some step in the session, when the user visits a webpage with content relevant to her ad. The number of slots sought by the advertiser, as well as his value from having his request granted, varies by the combination of advertiser and user. As advertiser requests arrive online (each triggered by the current state of the user's browsing history), an *ad server* must decide which requests to grant, since at any time only a single advertiser can have access to the user. The objective of the ad server is to maximize total value on a user. For our purposes, it suffices to consider the decisions made by the ad server for a single user, since the overall value can be summed over the individual users. Additionally, a user may exit his browsing session at any step – a classical assumption in web user modeling. Thus, the ad server faces a tradeoff between advertiser requests consisting of short sequences with a high average value and longer ones of high total value. The prototypical decision for the ad server then becomes: for each request, decide whether and when to serve the request, and whether to pre-empt a slot sequence currently in progress from a prior request.

1.1 Model

We abstract the following online job scheduling problem from this setting. At each step, the user stops surfing with probability $1-\beta$ (so the surfing time is a geometric random variable). Jobs, which correspond to advertisers' stories, arrive online. A job s has an arrival time a_s, a length ℓ_s and a per-unit (or per impression) value v_s. (In general, the per-unit values need not be the same for all ads in a job; we discuss this in Section 4.) An input sequence Q is a set of jobs $\{(v_s, a_s, \ell_s)\}$, and corresponds to the set of advertisers that arrive as the user surfs pages relevant to them. Multiple realistic assumptions can be made about value derived from suspending and restarting interrupted jobs – in this paper we assume that such suspension is not allowed. That is, jobs can be interrupted, but once interrupted, no value can be obtained from its remaining units.

We investigate two models that differ in the maximum delay that can be allowed between the arrival and scheduling of a job, corresponding to two natural models of storyboard a. In the *no deadlines* model, a job s can be scheduled anytime after its arrival (here, advertisers are willing to advertise to a user at any time $t \geq a_s$ after discovering that the user is relevant). In the *sharp deadlines* model, a job s is lost unless scheduled immediately upon arrival at $t = a_s$ (here, advertisers lose interest in the user as soon as she navigates away from the relevant page).

[1] In 2002 [11], the New York Times offered advertisers the opportunity to completely control the ads seen by a particular user for a small number (4-8) of consecutive pages viewed by this user – the first instance of online storyboarding.

A feasible schedule is one in which all scheduled units of a job are scheduled consecutively, subject to the deadline constraint. The discounted reward $V_{\mathcal{S}}$ from a feasible schedule $\mathcal{S}$ is its expected value $\sum_{t=0}^{\infty} \beta^t v(t)$, where $v(t)$ is the per-unit value of the job in progress at time t. We want to design an online scheduler whose discounted reward is competitive against the discounted reward of the (optimal) offline scheduler that knows the entire input sequence in advance. [2]

1.2 Results and organization

We show that there is a *sharp separation* in the power of an online scheduler that has no deadlines versus one with sharp deadlines: we give a constant competitive algorithm for the first case, whereas we show that no randomized online algorithm can achieve better than a polylogarithmic ratio in the second. Note that while an online scheduler with no deadlines can clearly obtain higher reward than an online scheduler with sharp deadlines, the offline scheduler is correspondingly advantaged as well, so it is not a priori obvious how the competitive ratios in the two models will compare.

- In the no-deadline model, we give a natural deterministic algorithm that is 7-competitive against the offline scheduler (Section 2).

  If the discount factor $\beta = 1$, the algorithmic problem in the no deadline model is trivial: never preempt. For any $\beta < 1$, there is a *preemption-delay* tradeoff – the scheduler either has to preempt the current job and lose all its remaining value, or has to delay the newly arrived jobs and pay a (nonzero) delay cost due to the discount factor. (Observe that preemption is "cheap" for β near 0 and delay is "expensive", whereas the converse is true for β close to 1.) In fact, the problem has a *discontinuity* at $\beta = 1$ – while at $\beta = 1$, the trivial (online) algorithm that never preempts is optimal, no deterministic algorithm can have a competitive ratio better than $(2 - 3\epsilon)$ for $\beta = 1 - \epsilon$, $\epsilon > 0$, as the following example shows. The first job arrives at time 0 with per-unit value 1 and infinite length. If the deterministic online algorithm begins this job at time t, the adversary introduces a job at time $t + 1$ with value $1/(1 - \beta)$ and length 1. The offline scheduler gets a total discounted reward of $\frac{\beta^t(\beta+\beta^2)}{1-\beta}$. The deterministic algorithm can get no more than $\max\left(\beta^t + \beta^{t+1}\frac{1}{1-\beta}, \frac{\beta^t}{1-\beta}\right) = \frac{\beta^t}{1-\beta}$, which gives a competitive ratio of $\beta + \beta^2 \geq 2 - 3\epsilon$ for $\beta = 1 - \epsilon$. The intuition behind the discontinuity is that for any $\beta < 1$, there is always a job with high enough value that makes preemption worthwhile; this is not true for $\beta = 1$.

- In the sharp deadlines model, we adapt the proof from [3] to show that despite partial credits and the discount factor, no online scheduler can achieve a reward better than a polylogarithmic factor of the offline scheduler (Section 3).

- In Section 4 we consider a natural extension from the storyboarding application, with increasing (rather than constant) per-unit values. For the extreme case where value is obtained only when the job is finished, even with no deadlines, no online deterministic algorithm can achieve any constant competitive factor with respect to the offline scheduler. However, we show that constant competitiveness can be obtained when the lengths of the jobs are bounded, to give a logarithmic approximation.

[2] Note that the offline solution does not know exactly when the user stops surfing, but only maximizes expected value. An online scheduler is very limited with respect to the offline scheduler which knows the stopping time of the user. These limitations are also observed in other online settings [1].

1.3 Related Work

See [6] for a survey of the vast literature on job scheduling. The main difference with our work is the infinite horizon with discount factor β. Our main algorithmic results are in the no-deadline model, which is trivial for $\beta = 1$ and has consequently not received attention in the literature. Woeginger [12] studies the model closest to ours in terms of the feasibility of an allocation – this is, in fact, the sharp deadline model with no partial credits at $\beta = 1$. However, the algorithmic results in [12] are under restrictions on the input that are inapplicable in our setting; the hardness results are for deterministic algorithms for $\beta = 1$ and no partial credits in the sharp deadline model. As mentioned in Section 1.2, Cannetti and Irani [3] study lower bounds in the model with sharp deadlines and no partial credit at $\beta = 1$. There has also been work in mechanism design which involves online allocation problems [4, 9, 7, 5]. These models differ from ours in multiple ways, most notably in not having a time-discounted infinite horizon. The discounted reward scenario has been considered previously in settings other than job-scheduling [2].

2 A greedy algorithm for the no-deadline model

We now present a greedy algorithm for online job scheduling in the no-deadline model. There are two aspects to consider – the *preemption-delay* tradeoff (since jobs cannot be resumed after interruption), and the fact that jobs arrive online. In fact, the preemption-delay tradeoff is also faced by an offline scheduler; we show in Appendix 5 that the offline problem is NP-complete, by a reduction from subset-sum. The online arrival of jobs further compounds matters – in the no-deadline case the online algorithm must decide whether to schedule a long job, or wait for imminent high per-unit value jobs. When all jobs are available at time 0, the problem is easy:

Lemma 1. *If all jobs are present at time* 0*, scheduling jobs in the decreasing order of their per-unit values is optimal.*

For a set of jobs S, let $V(S)$ denote the reward obtained by scheduling jobs in S as in Lemma 1: in computing $V(S)$ we pretend that all jobs in S have arrival time 0.

In general, of course, there will be new arrivals while a job is in progress. At time t, let s be the currently scheduled job with value v_s and l'_s remaining unscheduled units. Let A_t be the set of all jobs with value higher than v_s. The greedy decision at time t is based on comparing the reward from preempting s and scheduling jobs in A_t immediately, to the reward from completing s and then scheduling A_t. That is, we preempt s if

$$\frac{1-\beta^{l'_s}}{1-\beta} v_s + \beta^{l'_s} V(A_t) < V(A_t).$$

Rearranging the above inequality gives us the following rule for preemption:

$$v_s < (1-\beta)V(A_t).$$

This rule simply compares the benefit from scheduling another unit of the current job, to the cost of delaying the jobs in A_t by one step. The corresponding greedy algorithm, $\mathcal{G}$, is given in Figure 1. Note that A_t now only contains jobs that arrive *after* the current job is scheduled by $\mathcal{G}$.

Algorithm $\mathcal{G}$:

At each time t:
If no job is in progress,
Schedule a job with the highest per-unit value.
Otherwise, let s be the current job.
Let A_t be the set of all jobs with per-unit value higher than v_s.
If $v_s < (1-\beta)V(A_t)$ then
Preempt s
Schedule a job with the highest per-unit value.
Otherwise, continue with the current job.

Figure 1: A greedy algorithm for the no-deadline model

2.1 Analysis

In this section, we prove that $\mathcal{G}$ is 7-competitive against the offline scheduler (Theorem 1). In fact, we prove that $\mathcal{G}$ is 7-competitive against a stronger optimal scheduler $\mathcal{O}$ which is allowed to resume jobs after interruption. The stronger scheduler is equivalent to modifying the input sequence Q to replace each job (v_s, a_s, ℓ_s) by ℓ_s jobs with length 1 and value v_s: the optimal scheduler $\mathcal{O}$ greedily schedules the available unit with highest value at each time. For simplicity of presentation, we assume that all values v_s are distinct[3].

We denote by $\mathcal{A}(\mathcal{Q})$ the schedule returned by a scheduler $\mathcal{A}$ on an input Q, and by $V_{\mathcal{A}(\mathcal{Q})}$ the discounted reward from this schedule. When it is clear from the context, we use $\mathcal{A}$ to denote both the schedule and the scheduler.

There are two factors that make $\mathcal{G}$'s schedule suboptimal: preempted units that are never scheduled, and units that are delayed. To analyze these, we first introduce a new input sequence Q'' with suitably delayed arrivals, as defined below. The idea behind the construction of Q'' is that, roughly speaking, $\mathcal{G}$'s schedule is the same on Q and Q'', and $\mathcal{G}$ is suboptimal with respect to $\mathcal{O}(Q'')$ due only to preempted units. In Lemma 6, we show $V_{\mathcal{O}(Q'')} \le 2V_{\mathcal{G}}$. To account for the delay cost, we now only need to relate $V_{\mathcal{O}(Q)}$ to $V_{\mathcal{O}(Q'')}$ – we do this by introducing another input sequence Q', with some units arriving earlier than in Q, such that $V_{\mathcal{O}(Q')} \ge V_{\mathcal{O}(Q)}$. We then relate $V_{\mathcal{O}(Q')}$ to $V_{\mathcal{O}(Q'')}$ through Lemmas 2, 3, 4, and 5 to get the result in Corollary 1.

Note that with respect to the scheduler $\mathcal{O}$, we can think of Q as a sequence of units instead of a sequence of jobs. The arrival of the k'th unit of job s in Q is defined as $a_s + k - 1$. For job s, let r_s be the time at which $\mathcal{G}$ begins s, and e_s be the time at which the last unit of s is scheduled, i.e., s is completed at e_s or preempted at $e_s + 1$. Also, define D_s to be a set of units such that each unit $i \in D_s$ satisfies the following properties:

1. $\mathcal{O}(Q)$ schedules i sometime between r_s and e_s.

2. $v_i > v_s$.

3. If i belongs to job s', then $a'_s > r_s$.

[3]One can perturb all values with arbitrary small noise.

Now, based on the schedule $\mathcal{G}$ for the input Q, we construct two sequences with modified arrivals, denoted by Q' and Q''. If unit i belongs to D_s for a job s, then i arrives in Q' at time r_s. Also, the arrival of i arrives in Q'' at time e_s+1. If i does not belong to D_s for any s, then its arrival time is the same in Q, Q' and Q''. Let $\mathcal{O}(Q')$ and $\mathcal{O}(Q'')$ denote the schedules of $\mathcal{O}$ on the input sequences Q' and Q''. Also, let r'_i (resp. r''_i) be the time at which i is scheduled in $\mathcal{O}(Q')$ (resp. $\mathcal{O}(Q'')$).

Lemma 2. *If unit i does not belong to any set D_s, then $r''_i \leq r'_i$.*

Proof. We prove the lemma by contradiction. Assume i is the first unit scheduled in $\mathcal{O}(Q')$ such that $r'_i < r''_i$. Let j be the unit scheduled in $\mathcal{O}(Q'')$ at time r'_i. If i does not belong to any set D_s, then it arrives at the same time in Q' and Q''. Therefore, for input Q'' and at time r'_i, $\mathcal{O}$ could have scheduled unit i. Hence, we have $v_j \geq v_i$. Consider the following cases.

1. Suppose $v_j > v_i$. Note that the arrival of every unit is Q' is no later than its arrival in Q''. Also, $\mathcal{O}$ always schedules a unit with the highest value. Therefore, $r'_j < r'_i = r''_j$, which contradicts the assumption that i is the first unit such that $r'_i < r''_i$.
2. If $v_j = v_i$, then these units belong to the same job. Since $\mathcal{O}$ first schedules the unit with the earlier arrival, we have $r'_j < r'_i = r''_j$, which again leads to a contradiction.

□

The lemma above gives us the following inequality.

$$V_{\mathcal{O}(Q')} \leq \sum_s \sum_{i \in D_s} \beta^{r'_i} v_i + V_{\mathcal{O}(Q'')} \tag{1}$$

We show that $\sum_s \sum_{i \in D_s} \beta^{r'_i} v_i \leq V_{\mathcal{G}} + 2V_{\mathcal{O}(Q'')}$. Let $l_s = e_s - r_s + 1$ denote the number of units of job s scheduled by $\mathcal{G}$. Note that

$$\sum_s \sum_{i \in D_s} \beta^{r'_i} v_i = \sum_s (1-\beta^{l_s}) \sum_{i \in D_s} \beta^{r'_i} v_i + \sum_s \beta^{l_s} \sum_{i \in D_s} \beta^{r'_i} v_i$$

We prove the claim by bounding each of term in the RHS above.

Lemma 3. $\sum_s (1-\beta^{l_s}) \sum_{i \in D_s} \beta^{r'_i} v_i \leq V_{\mathcal{G}}$

Proof. Recall that A_t is the set of all jobs that are available at time t and have per-unit value higher than the current job. Let $t = e_s$. Observe that all units in D_s belong to the jobs in A_t. Therefore, $V(A_t) \geq V(D_s)$. Because the algorithm did not preempt job s at time t, we have

$$v_s \geq (1-\beta) V(A_t) \geq (1-\beta) V(D_s).$$

Therefore,

$$\frac{1-\beta^{l_s}}{1-\beta} v_s \geq (1-\beta^{l_s}) V(D_s).$$

Summing up this inequality over all jobs s, we have

$$V_{\mathcal{G}} \geq \sum_s (1-\beta^{l_s}) \sum_{i \in D_s} \beta^{r'_i} v_i.$$

□

Now, we find a matching between units in D_s and units in $\mathcal{O}(Q'')$. Define m_i to be the *unique* slot i is matched to. The matching is constructed as follows. Let U_t be the sorted list of all unmatched units up to time t; U_0 is initialized to be the empty set. At each time t, add all unmatched units from U_{t-1} plus all units i such that $i \in D_s$ and $r'_i + l_s = t$ to U_t. Then, let j be the unit with the highest value in U_t. Ties are broken in favor of units that arrive earlier in Q. Finally, let $m_j = t$.

Lemma 4. *For unit $i \in D_s$, we have $m_i \leq r''_i$.*

Proof. First observe that for $i \in D_s$, $r''_i \geq r'_i + l_s$. We prove the lemma by induction on the scheduling time r''_i in $\mathcal{O}(Q'')$ of a unit $i \in U_t$. The basis of the induction is trivial. Now, suppose for all units j, $r''_j < r''_i$, we have $m_j \leq r''_j$. Suppose i has not been matched yet. Hence, i belongs to $U_{r''_i}$. Observe that by induction hypothesis, all unmatched units in $U_{r''_i}$ are available to $\mathcal{O}(Q'')$. Since $\mathcal{O}(Q''))$ schedules i at this time (r''_i), i has the earliest arrival among available units with the highest value. Therefore, i also has the earliest arrival among units with the highest value in $U_{r''_i}$. Thus, i would be matched at this step and we have $m_i = r''_i$. $\square$

By the lemma above we have,

$$\sum_s \sum_{i \in D_s} \beta^{m_i} v_i \leq V_{\mathcal{O}(Q'')} \tag{2}$$

Lemma 5. $\sum_s \sum_{i \in D_s} (\beta^{r'_i + l_s} - \beta^{m_i}) v_i \leq V_{\mathcal{O}(Q'')}$

Proof. Consider unit $j \in U_t$. If no other unit is added to U_t, then j would be matched at time $t + \pi_j - 1$, where π_j is the position of j in the sorted list U_t. Now suppose unit i is added to U_t at this point (units are added to U_t in the order of increasing r'_i). Then, the position to which j would be matched increases by 1. Let n_{ij} be the slot that unit j would be matched to right *before* i is added. Also, let n'_{ij} be the slot that unit j would be matched to right *after* i is added. Note that $n'_{ij} - n_{ij} \leq 1$. It is easy to see that we have

$$\sum_s \sum_{i \in D_s} (\beta^{r'_i + l_s} - \beta^{m_i}) v_i = \sum_s \sum_{i \in D_s} \sum_j (\beta^{n_{ij}} - \beta^{n'_{ij}}) v_j$$

Consider unit $i \in D_s$. Let i^* be the unit that is scheduled in $\mathcal{O}(Q'')$ at time r'_i. Suppose for unit j we have $n'_{ij} - n_{ij} = 1$. Observe that by Lemma 4, we have $v_{i^*} \geq v_j$. Hence,

$$\sum_s \sum_{i \in D_s} \sum_j (\beta^{n_{ij}} - \beta^{n'_{ij}}) v_j \leq \sum_s \sum_{i \in D_s} \sum_j (\beta^{n_{ij}} - \beta^{n'_{ij}}) v_{i^*}$$

Also, note that for all units j such that $n'_{ij} - n_{ij} = 1$, all values of n_{ij}'s are unique, and are greater than or equal to $r'_i + l_s$. Therefore, we have

$$\sum_s \sum_{i \in D_s} \sum_j (\beta^{n_{ij}} - \beta^{n'_{ij}}) v_{i^*} \quad \leq \quad \sum_s \sum_{i \in D_s} \sum_{t \geq r'_i + l_s} (1 - \beta) \beta^t v_{i^*} = \sum_s \sum_{i \in D_s} \beta^{r'_i + l_s} v_{i^*} \leq \sum_s \sum_{i \in D_s} \beta^{r'_i} v_{i^*}$$

But, by definition of i^*, $\sum_s \sum_{i \in D_s} \beta^{r'_i} v_{i^*} \leq V_{\mathcal{O}(Q'')}$. Therefore,

$$\sum_s \sum_{i \in D_s} (\beta^{r'_i + l_s} - \beta^{m_i}) v_i \leq V_{\mathcal{O}(Q'')}$$

$\square$

Putting all lemmas above together we get a bound on the delay cost of $\mathcal{G}$:

Corollary 1. $V_{\mathcal{O}(Q')} \leq V_{\mathcal{G}} + 3V_{\mathcal{O}(Q'')}$

Next we handle the preemption cost.

Lemma 6. $V_{\mathcal{O}(Q'')} \leq 2V_{\mathcal{G}}$.

Proof. Note that every unit in $\mathcal{O}(Q'')$ has the following property: either $\mathcal{G}$ schedules it no later than $\mathcal{O}$, or $\mathcal{G}$ never schedules it at all, i.e., the unit is preempted. Hence, we complete the proof of the lemma by showing that the value of the preempted units is bounded by $V_{\mathcal{G}}$.

A *preemption chain* $(s_1, \cdots, s_n)$ is a sequence of jobs defined by the following properties. (i) s_1 does not belong to any previous chain. (ii) Story s_i, $1 \leq i \leq n$, is preempted at time t_i. (iii) Let k_i be the sum of lengths of the jobs available at time t_i that have higher per-unit value than s_i. For each $1 \leq i < n$, $t_i < t_{i+1} \leq t_i + k_i - 1$. (iiii) $\mathcal{G}$ does not preempt any job between t_n and $t_n + k_n - 1$. Because the number of jobs is finite, chains are of finite length.

Consider a chain $(s_1, \cdots, s_n)$. Let A_n be the jobs with higher per-unit value than s_n that are available at time t_n. Since s_n is preempted, we have $(1-\beta)V(A_n) > v_{s_n}$. Also, because there is no preemption between t_n and $t_n + k_n - 1$, it is easy to see that the reward $\mathcal{G}$ obtains from the units scheduled during this time is at least

$$\beta^{t_n} V(A_n) \geq \beta^{t_n}(v_{s_n}/(1-\beta)) \tag{3}$$

Note that the per-unit value of the jobs is increasing along a chain. Therefore, in $\mathcal{O}(Q'')$ all preempted units of $s_1, \cdots, s_n$ are scheduled after $t_n + k_n - 1$. Also, observe that the total reward $\mathcal{O}$ could obtain from these jobs is $\beta^{t_n+k_n}(v_{s_n}/(1-\beta))$. Plugging into (3) we get

$$\beta^{t_n+k_n}(v_{s_n}/(1-\beta)) \leq \beta^{t_n}(v_{s_n}/(1-\beta)) \leq \beta^{t_n} V(A_n)$$

The lemma follows from summing this inequality over all chains. □

Theorem 1. *Algorithm $\mathcal{G}$ is 7-competitive against the offline scheduler.*

Proof. Combining the results in Corollary 1 and Lemma 6, and using the fact that $V_{\mathcal{O}(Q)} \leq V_{\mathcal{O}(Q')}$, we get the above claim. □

The lower bound of 2 for $\beta \to 1$ (Section 1) can be exhibited very simply for this algorithm, with one job of infinite length and value $v_1 = 1$ arriving at time 0, and a second job of length 1 and value $\frac{1}{1-\beta}$ arriving at time 1.

3 A lower bound for the sharp-deadlines model

We now derive a lower bound on the competitiveness of any randomized online algorithm in the model with sharp deadlines. Our construction is the same as that in [3]. However, our proofs require more effort due to partial credits and the discount factor; as opposed to [3], the online algorithm can now potentially utilize the partial credits to get closer to optimal, and the offline scheduler itself becomes less advantaged in the presence of a discount factor, simply because the value from the future gets discounted (e.g., when $\beta \to 0$, the greedy algorithm is optimal). We show that, when β is at least 3/4, despite partial credits and the discount factor, we can derive a poly-logarithmic lower bound on the performance of any randomized online algorithm in the sharp-deadlines model. The underlying intuition is that partial credits and the discount factor only add lower order terms to the reward obtained by the online algorithm, in the lower-bound construction of [3].

Let $v_{\max}$ and $v_{\min}$ denote the maximum and minimum per-unit values, and $L_{\max}$ and $L_{\min}$ be the maximum and minimum lengths of the jobs. Define $\mu = \max\left(\frac{L_{\max}}{L_{\min}}, \frac{v_{\max}}{v_{\min}}\right)$. We construct a family of examples for which no randomized online algorithm can achieve competitiveness better than $\Omega\left(\sqrt{\frac{\log\mu}{\log\log\mu}}\right)$ on these examples.

The construction of the lower bound is described in Appendix 6. It consists of a sequence of oblivious adversaries, $\mathtt{ADV}_i$, defined recursively. Intuitively, adversary $\mathtt{ADV}_i$ acts as follows: it orders new jobs of type i; also if at any time the probability of the online algorithm working a job of type i is higher than a certain threshold, it calls $\mathtt{ADV}_{i-1}$ to obtain jobs of shorter length but higher per-unit value. The threshold function and the formal strategy of $\mathtt{ADV}_i$ are defined in the Appendix, where we also give the proof of the following theorem:

Theorem 2. *When $\beta \geq 3/4$, given any randomized preemptive scheduler, for the problem instance generated by the family of adversaries $\mathtt{ADV}_k$, the competitiveness is at best $\Omega\left(\sqrt{\frac{\log\mu}{\log\log\mu}}\right)$.*

Observe that if the per-unit value of all jobs differ by at most a constant factor λ, then the algorithm that preempts the current story only if the new job is longer, is λ-competitive. Therefore, using standard techniques, one can design a $O(\log\frac{v_{\max}}{v_{\min}})$-competitive algorithm, see Section 4.2.

4 Increasing per-unit values

In the storyboarding application it is natural for advertisers to have increasing values for the ads displayed in their sequence, particularly when the sequence of ads form a story. We now consider the extreme case where advertisers derive value only when their entire story is shown without interruption. Formally, each job (story) s is specified by a length ℓ_s and a final value f_s. Note that the value obtained from the first $\ell_s - 1$ units of the job is zero. The total value of job s (if started at time zero) is equal to $\beta^{\ell_s - 1} f_s$. We focus here on the no-deadline model, since the sharp-deadlines model with these valuations is similar to the model studied in [12, 3], despite the discount factor.

4.1 A lower bound

We first show that no deterministic algorithm can be constant competitive in this model. For every c we construct an input sequence consisting of one long job and several short jobs, such that preempting the long job at any step would lead to a ratio worse than c, and finishing the job leads to a bad competitive ratio as well.

Theorem 3. *For any constant $c > 0$, no deterministic algorithm can achieve a competitive ratio of c with respect to the offline scheduler* OPT*.*

Proof. Suppose a deterministic algorithm claims a ratio c. Consider an input in which there is a job of length L (specified later) and total value $V = 1$, arriving at time 0. A stream of jobs of length 1 each and value α/β^{i-1} arrives from time 1 onwards, until the deterministic algorithm preempts the current job. If the deterministic algorithm never preempts the long job, the arrivals never stop. The value of α is a function of c and will be chosen later. With these arrivals, at time i, OPT obtains a reward of $\beta^{i-1}\frac{\alpha}{\beta^{i-1}} = \alpha$, so that its total value grows linearly with time.

We choose α to ensure that preempting at any point causes the deterministic algorithm to get value below c times what OPT has already got so far: at time l, let $OPT(l)$ denote the total value that OPT gets using arrivals so far (including the long job); let $V_p(l)$ denote the value from preempting the long job at time l;

$$V_p(l) = \beta^{l-1}\alpha\left(\frac{1}{\beta^{l-1}} + \frac{\beta}{\beta^{l-2}} + \cdots + \beta^{l-1}\right) = \alpha(1 + \beta^2 + \cdots + \beta^{2l-2}) = \frac{\alpha(1-\beta^{2l+2})}{1-\beta^2}.$$

Then, since $V = 1$,

$$\frac{OPT(l)}{V_p(l)} = \frac{\alpha l + \beta^l V}{\alpha\frac{1-\beta^{2l+2}}{1-\beta^2}} = \frac{l(1-\beta^2)}{1-\beta^{2l+2}} + \frac{\beta^{l+1}(1-\beta^2)}{\alpha(1-\beta^{2l+2})} \tag{4}$$

We want to choose α such that this is greater than c for all l. For $l > c/(1-\beta^2)$, (4) is greater than c, since the second term is positive. For smaller values of l, we choose $\alpha(c)$ to satisfy $\beta^{\frac{c}{1-\beta^2}+1}(1-\beta^2) > \alpha c$. This ensures that for all values of l, preempting the long job leads to an approximation factor that is worse than c. We choose L to be large enough to ensure that the approximation factor on delaying is also worse than c, *i.e.*, such that

$$\frac{OPT(L)}{V_d} = \frac{\alpha L + \beta^l}{1 + \alpha\frac{1-\beta^{2l+2}}{1-\beta^2}} > c.$$

□

4.2 A competitive algorithm

Now we present an algorithm for the case when the value of a job is obtained only after its last unit is scheduled. Our algorithm is $O(\log \frac{f_{\max}}{f_{\min}})$-competitive with respect to the offline scheduler, where $f_{\max} = \max_s\{f_s\}$ and $f_{\min} = \min_s\{f_s\}$.

We first consider a special case of the model, in which all the values of f_s are equal to 1. Observe that if $\beta \leq \frac{1}{2}$ then the algorithm that always preempts in the favor of a job that finishes earlier achieves a competitive ratio of $\frac{1}{1-\beta} \geq 2$. For $\beta \geq \frac{1}{2}$, we propose the following algorithm, called $\mathcal{C}$: If there is no job in progress, schedule a job of the shortest length. Otherwise, let s be the current job. Upon the arrival of a new job j, preempt s and schedule j if

$$\frac{\ell_s}{\ell_j} > \frac{1}{1-\beta}$$

Lemma 7. *For $\beta \geq \frac{1}{2}$, algorithm $\mathcal{C}$ is $12\frac{2e-1}{e-1}$-competitive with respect to the offline scheduler OPT.*

Proof. Most parts of the proof are similar to Theorem 1 and, therefore, are discussed briefly. First assume the lengths of all jobs are bounded by $\frac{1}{1-\beta}$. This assumption will be removed later.

For job s, define D_s to be the set of units that are scheduled by OPT at the time $\mathcal{C}$ schedules s. Let r'_i denote the time when OPT schedules unit i. Similar to Theorem 1, we find a matching from units in D_s to units scheduled by $\mathcal{C}$. Note that this optimal solution must schedule jobs without interruption; we refer to units only for convenience.

Let U be the list of all unmatched jobs, initialized to be the empty set. Each unit i, in a set D_s, is added to U at time $r'_i + \ell_s$. At each time t, units are removed from U in two ways: i) unit i is displayed by $\mathcal{C}$ at time t. ii) unit i has the lowest r'_i in U. In both of the cases above, i is removed from U and we have $m_i = t$.

By definition, we have $m_i \leq r_i$, where r_i is the time that $\mathcal{C}$ schedules unit i. Also, for every t, there are at most two units i and j such that $m_i = m_j = t$. Note that we can replace each job of length l with another job of per-unit value v_s such that $\frac{1-\beta^{\ell_s}}{1-\beta} v_s = \beta^{\ell_s}$. Because neither $\mathcal{C}$ nor OPT preempt any jobs, these replacements do not change the values of the schedules. Therefore,

$$\sum_i \beta^{m_i} v_i \leq 2V_{\mathcal{C}} \tag{5}$$

Observe that after the replacements, at time t, the maximum value of a unit in U is at most the value of the unit scheduled by $\mathcal{C}$ at time t. The reason is that units in U belong to the stories that are available to $\mathcal{C}$ (or are currently being displayed); and $\mathcal{C}$ always chooses a job with the highest value-per-unit. Therefore, similar to Lemma 5, we have that

$$\sum_s \sum_{i \in D_s} (\beta^{r'_i + \ell_s} - \beta^{m_i}) v_i \leq V_{\mathcal{C}} \tag{6}$$

Also, because all lengths are bounded by $\frac{1}{1-\beta}$, for $\beta \geq \frac{1}{2}$, we have $\beta^{\ell_s} \geq \beta^{\frac{1}{1-\beta}} \geq \frac{1}{4}$. Therefore,

$$\sum_s \sum_{i \in D_s} \beta^{r'_i + \ell_s} v_i \geq \frac{1}{4} V_{\mathrm{OPT}} \tag{7}$$

Now we remove that assumption that job lengths are bounded. For each length l, the total reward that could be obtained from all jobs of length $\geq \frac{l}{1-\beta}$ is at most

$$\beta^{\frac{l}{1-\beta}-1} \frac{1}{1-\beta^{\frac{l}{1-\beta}}} \leq \beta^{\frac{l}{1-\beta}-1} \frac{1}{1-\beta^{\frac{1}{1-\beta}}} \leq \left(\frac{1}{1-\frac{1}{e}}\right) \beta^{\frac{l}{1-\beta}-1}$$

Therefore, if the algorithm obtains the value of a job of length l, it can discard all jobs of length $> \frac{l}{1-\beta}$, and still be constant competitive with respect to the offline scheduler. Now by plugging in (5), (6), and (7), it is easy to see that this algorithm is $12\frac{2e-1}{e-1}$ competitive.

□

We can build on the above argument to give a $O(\log \frac{f_{\max}}{f_{\min}})$-competitive algorithm, $\mathcal{A}$, for the no-partial-credit case. The argument is rather standard. We first partition the jobs in the optimal solution into buckets based on f_s into buckets $[2^i, 2^{i+1})$. There are $\log(f_{\max}/f_{\min})$ such buckets. Let O_i be the value that the optimal solution gets from jobs in bucket i, $O = \sum_i O_i$. We randomly choose a bucket j. The previous argument shows us that restricting the input to jobs in this bucket gives us value at least $O_j \frac{e-1}{12(2e-1)}$, since the optimal solution with this input is at least O_j. Now, the expected value we get is

$$\sum_i \frac{1}{\log(f_{\max}/f_{\min})} \frac{e-1}{12(2e-1)} O_i \geq \frac{e-1}{12(2e-1)} \frac{\sum_i O_i}{\log(f_{\max}/f_{\min})} = \frac{e-1}{12(2e-1)} \frac{O}{12\log(f_{\max}/f_{\min})}.$$

Theorem 4. *Algorithm $\mathcal{A}$ is $O(\log(f_{\max}/f_{\min}))$-competitive with respect to the offline scheduler in the model with no deadlines and no partial credits.*

References

[1] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson, On the Power of Randomization in On-Line Algorithms. *Algorithmica*, 11(1): 2-14, 1994.

[2] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. *SIAM Journal of Computing*, 37(2):653–670, 2007.

[3] R. Canetti and S. Irani. Bounding the power of preemption in randomized scheduling. *SIAM Journal of Computing*, 27(4):993–1015, 1998.

[4] M. T. Hajiaghayi, R. Kleinberg, M. Mahdian, and D. Parkes. Online auctions with re-usable goods. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 165–174, New York, NY, USA, 2005.

[5] M. T. Hajiaghayi, R. Kleinberg, and D. C. Parkes. Adaptive limited-supply online auctions. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 71–80, New York, NY, USA, 2004.

[6] A. W. Kolen, J. K. Lenstra, C. H. Papadimitriou, and F. C. Spieksma. Interval scheduling: A survey. *Naval Research Logistics*, 54:530–543, 2007.

[7] R. Lavi and N. Nisan. Online ascending auctions for gradually expiring items. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1146–1155, Philadelphia, PA, USA, 2005.

[8] marketingterms.com. Surround sessions.
http://www.marketingterms.com/dictionary/surround_session.

[9] R. Porter. Mechanism design for online real-time scheduling. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 61–70, New York, NY, USA, 2004.

[10] PricewaterhouseCoopers. IAB internet advertising revenue report.
http://www.iab.net/media/file/IAB_PwC_2007_full_year.pdf.

[11] N. Y. Times. What is surround sessions?
http://www.nytimes.com/ads/marketing/surroundsessions.

[12] G. J. Woeginger. On-line scheduling of jobs with fixed start and end times. *Theor. Comput. Sci.*, 130(1):5–16, 1994.

5 The offline problem in no-deadline model is NP-Hard

Proposition 1. *The problem of computing the optimal schedule of jobs, even when all arrivals are known in advance, is NP-complete.*

Proof. The proof is by reduction from subset-sum. The subset-sub problem is defined as follows: given a set S of integers, and a sum b, is there a subset of S whose sum is b?

Given an instance of subset sum, for each $n_s \in S$, define a job s of arrival time 0, $v_s = 1$, and length n_s . Also, define a job of arrival time b and value M, where M is chosen large enough so that the optimal schedule has to display M at b. If there is a subset of S with sum b, then the value of the optimal schedule is

$$\sum_{t=0}^{b-1} \beta^t + M\beta^b + \sum_{t=b+1}^{\sum s_i} \beta^t$$

If there is no such subset, the reward of the offline scheduler is strictly less than this value. □

6 Lower bound for the sharp-deadlines model

Our construction follows that of [3]. The proofs of the corresponding lemmas differ. For completeness, we still outline the construction, while pointing out the major differences. We define a sequence of adversaries $\mathtt{ADV}_i$ recursively. $\mathtt{ADV}_i$, when deployed against the randomized algorithm, creates a family of examples giving a lower bound of γ_i where γ_i is defined in equation (8). Adversary $\mathtt{ADV}_i$ uses $i+1$ types of jobs, the i^{th} job having a length of $L_i = k^{4i}$ and a per-unit value $v_i = k^{2-2i}$. Thus the total value $V_i = v_i \frac{1-\beta^{L_i}}{1-\beta}$. A online randomized algorithm can be defined by a time varying probability distribution $\{p_i(t)\}$ where $p_i(t)$ is the probability that the algorithm is working on a job of type i at time t. We assume that the adversary has access to the probability distribution $\{p_i(t)\}$ for every time step, but not to the actual coin-tosses of the algorithm. Also, we assume that the discount factor $\beta \geq 3/4$.

Intuitively, adversary $\mathtt{ADV}_k$ acts as follows: it orders a new job of type k every ℓ_k steps; also at every step, if the probability of the online algorithm working a job of type k is higher than a certain threshold, it calls $\mathtt{ADV}_{k-1}$ to obtain jobs of shorter length but higher per-unit value. The threshold function for $\mathtt{ADV}_i$ is defined below, and the strategy of $\mathtt{ADV}_i$ is presented in Figure 2. The threshold function f_i is:

$$f_i(x) = \begin{cases} 1 - \alpha_i e^{\frac{\gamma_{i-1}}{V_i} x}, & \text{if } x \leq V_i; \\ 1 - \alpha_i e^{\gamma_{i-1}}, & \text{if } x > V_i. \end{cases}$$

where $\alpha_i = 2(1 - \gamma_i + \frac{2}{k^2})$ and

$$\gamma_i = \gamma_{i-1} \left(\frac{e^{\gamma_{i-1}}}{1 + \gamma_{i-1} e^{\gamma_{i-1}}} \right) + \frac{1}{k^2} \tag{8}$$

By Lemma 2 in [3], we have $\gamma_k \leq 4/\sqrt{k}$. Let $O(t)$ denote the maximum gain that can be accrued by the *jobs ordered up till time* t, appropriately discounted by the factor β. We have the same definitions of i-critical time units, and i-steps. For a k-step $\tau = [s_\tau, f_\tau]$ let $h_k(\tau)$ be defined as $O_{k+1}(s_\tau) - O_{k+1}(s_\tau - 1)$. Similarly, we can partition the set of jobs from the input sequence S

Adversary ADV_i**:**

At each time t:

If L_i divides t,
 Schedule a story of type i.
If $i > 0$ and $q_i(t) \geq f_i(O_i(t))$,
 call $\mathtt{ADV}_{i-1}$.

Figure 2: Strategy for $\mathtt{ADV}_i$

into *regular* and *non-regular* according to the definitions in [3], and can define $\mathbf{E}A_\tau(S)$ to be the reward that scheduler A can get in expectation from the *regular* jobs requested during the time interval τ in the request sequence S. We first bound the scheduler's gain from these regular jobs.

Lemma 8. *$\mathbf{E}A_\tau(S) \leq \gamma_k h_k(\tau)$, where γ_k is defined as in Equation 8.*

Proof. The inductive proof follows that of [3], Lemma 3. The cases for 0-th and 1-th adversary can easily be shown. For the k^{th} adversary, we assume a contradiction. We then construct a scheduler A' that mimics the scheduler A against the $(k-1)^{th}$ adversary. We then show that this violates the inductive hypothesis. The different cases are broken up as in [3] according to the nature of the interval τ; we only deal with cases that differ from those in [3].

Case 1. Step τ is not the first k-step. Case (1a) is trivial. For case (1b), we note that for each impression of type i, the probability that A' gets the value of the impression is $\frac{p_i(t)}{1-p_k(t)}$ which is $\frac{1}{1-p_k(t)}$ of the probability of A getting the impression. If this impression is from the job c with ending time e_c, we have that $p_k(t) \geq p_k(e_c)$, and thus $1 - p_k(t) \leq 1 - p_k(e_c) \leq \frac{\gamma_k}{\gamma_{k-1}}$, and thus the bound for case 1b holds as in the paper.

Case 2. In this case, τ is the first k-step, and thus the gain $k_k(\tau)$ is from the first k-type job that is ordered, at time 0. Thus, $h_k(\tau) = V_k$. We have to replace Equation 3.16 by the following

$$\mathbf{E}A_\tau(S) \leq V_k f_k(O_k(t_0)) + \sum_{i=1}^{l} \mathbf{E}A_{\tau_i}(S) + v_k \sum_{t=0}^{t_0} \beta^{t-1} p_k(t)$$

Arguing similarly, we get an equation like (3.17),

$$\begin{aligned}
&\mathbf{E}A_\tau(S) \\
&\leq V_k f_k(O_k(t_0)) + \sum_{i=1}^{l} (1 - p_k(e_{\tau_i}))\gamma_{k-1} h_{k-1}(\tau_i) + v_k \sum_{t=0}^{t_0} \beta^{t-1} p_k(t) \\
&\leq V_k f_k(O_k(t_0)) + \sum_{i=1}^{l} (1 - p_k(e_{\tau_i}))\gamma_{k-1} h_{k-1}(\tau_i) + v_k \sum_{t=0}^{t_0} \beta^{t-1}
\end{aligned}$$

By the definition of a k-step, if $|\tau|$ denotes the end of k-step τ, then $v_{k-1} \sum_{t=0}^{|\tau|} \beta^{t-1} = V_k$. Thus, in the above expression, since t_0 is either falls inside a k-step or is the very next step after it,

$v_k \sum_{t=0}^{t_0} \beta^{t-1} \leq V_k v_k / v_{k-1} \leq V_k / k^2$. Hence,

$$\mathbf{E} A_\tau(S) \leq V_k f_k(O_k(t_0)) + \sum_{i=1}^{l} (1 - p_k(e_{\tau_i})) \gamma_{k-1} h_{k-1}(\tau_i) + V_k/k^2,$$

and we need to show that the above expression is at most $\gamma_k V_k$. We show the proof in two parts. We write the above expression as $F + G$ where

$$F = V_k f_k(O_k(t_0)) + \sum_{i=1}^{l} (1 - p_k(s_{\tau_i})) \gamma_{k-1} h_{k-1}(\tau_i) \tag{9}$$

$$G = \sum_{i=1}^{l} (p_k(s_{\tau_i}) - p_k(e_{\tau_i})) \gamma_{k-1} h_{k-1}(\tau_i) + V_k/k^2 \tag{10}$$

We show that,

$$F \leq (\gamma_k - 2/k^2) V_k \tag{11}$$

and

$$G \leq 2V_k/k^2 \tag{12}$$

To start with,

$$\begin{aligned}
F &\leq V_k f_k(O_k(t_0)) + \sum_{i=1}^{l} (1 - p_k(s_{\tau_i})) \gamma_{k-1} h_{k-1}(\tau_i) \\
&\leq V_k f_k(O_k(t_0)) + \sum_{i=1}^{l} (1 - f_k(O_k(s_{\tau_i}))) \gamma_{k-1} h_{k-1}(\tau_i) \\
&\leq V_k f_k(O_k(t_0)) + \gamma_{k-1} \frac{1-\beta}{\beta \ln(1/\beta)} \int_0^{O_k(t_0)} (1 - f_k(y)) dy \\
&\leq V_k (1 - \alpha_k e^{\frac{\gamma_{i-1}}{V_k} O_k(t_0)}) + \gamma_{k-1} \frac{1-\beta}{\beta \ln(\frac{1}{\beta})} \int_0^{O_k(t_0)} \alpha_k e^{\frac{\gamma_{k-1}}{V_k} y} dy
\end{aligned}$$

after converting the sum to an integral and substituting the value for f_k in this range, since $x \leq V_k$ for the first k-step. Denote $x_0 = O_k(t_0)$. The above expression after integration becomes

$$\begin{aligned}
F &\leq V_k (1 - \alpha_k e^{\frac{\gamma_{k-1}}{V_k} x_0}) + \gamma_{k-1} \frac{1-\beta}{\beta \ln(\frac{1}{\beta})} \frac{V_k}{\gamma_{k-1}} (e^{\frac{\gamma_{k-1}}{V_k} x_0} - 1) \\
&\leq V_k \left(1 - \alpha_k \frac{1-\beta}{\beta \ln(\frac{1}{\beta})} + \alpha_k e^{\frac{\gamma_{k-1}}{V_k} x_0} \left(\frac{1-\beta}{\beta \ln(\frac{1}{\beta})} - 1 \right) \right)
\end{aligned}$$

Now, note that in this range, $e^{\frac{\gamma_{k-1}}{v_k} x_0} \leq e$. Also, $\frac{1-\beta}{\beta \ln(\frac{1}{\beta})} \geq 1$. So the above expression is at most $V_k \left(1 - \alpha_k \frac{1-\beta}{\beta \ln(\frac{1}{\beta})} + \alpha_k e \left(\frac{1-\beta}{\beta \ln(\frac{1}{\beta})} - 1 \right) \right)$. Now, for $\beta \geq 3/4$, $\frac{1-\beta}{\beta \ln(\frac{1}{\beta})} \leq \frac{e-0.5}{e-1}$. Thus, $F \leq$

$V_k(1-\alpha_k/2) = v_k(\gamma_k - \frac{2}{k^2})$ by definition of α_k, and hence we have proven Equation 11. For Equation 12, the exact argument as in the paper shows that $G \leq V_k/k^2 + V_k/k^2 = 2V_k/k^2$. Thus, we have the lemma.

□

Following Lemma 4 in [3], we know that the reward that the scheduler can obtain from the non-regular jobs is also bounded, since the value that the offline scheduler can extract out of such jobs is at most $\frac{1}{k}$ of the total optimal value. Thus, the reward that can be obtained by the our randomized scheduler is certainly bounded by $\frac{O_k}{k}$, which gives us the the following theorem, restated from Section 3.

Theorem 5. *When $\beta \geq 3/4$, for the problem instance generated by the family of adversaries $\mathtt{ADV}_k$, any randomized preemptive scheduler is at most $\Omega\left(\sqrt{\frac{\log\mu}{\log\log\mu}}\right)$ competitive.*

Proof. From the Lemma 8 in the appendix 6, if we can employ an adversary of the k^{th} level, then we can show that the randomized scheduler is at most $\gamma_k \leq 4/\sqrt{k}$ competitive against optimal for the request sequence that is generated by this adversary. Now, for this adversary, $\frac{v_{\max}}{v_{\min}} = k^{2k}$. Thus, if $m = k^{2k}$, we have that $\gamma_k \leq \frac{4}{\sqrt{k}} \leq \sqrt{\frac{\log\mu}{\log\log\mu}}$. Similarly, $\frac{L_{\max}}{L_{\min}} = k^{4k}$ also, giving us the factor in the claim. □