

Online Interval Scheduling: Randomized and Multiprocessor Cases

Stanley P. Y. Fung¹, Chung Keung Poon², and Feifeng Zheng³

¹ Department of Computer Science, University of Leicester, United Kingdom.
pyfung@mcs.le.ac.uk

² Department of Computer Science, City University of Hong Kong, China.
ckpoon@cs.cityu.edu.hk

³ School of Management, Xi'an JiaoTong University, China.
zhengff@mail.xjtu.edu.cn

Abstract. We consider the problem of scheduling a set of equal-length intervals arriving online, where each interval is associated with a weight and the objective is to maximize the total weight of completed intervals. An optimal 4-competitive algorithm has long been known in the deterministic case, but the randomized case remains open. We give the first randomized algorithm for this problem, achieving a competitive ratio of 3.618. We also prove a randomized lower bound of $4/3$, which is an improvement over the previous $5/4$ result, and a lower bound of 2 for a class of barely random algorithms which include our new algorithm. We also show that the techniques can be carried to the deterministic multiprocessor case, giving a 3.618-competitive 2-processor algorithm, a $5/4$ lower bound for any number of processors, and a 2 lower bound for 2 processors.

1 Introduction

We study the problem of scheduling a set of intervals which arrive online. Each interval has a weight and all intervals are of the same length. The objective is to schedule a set of non-overlapping intervals such that the total weight of all these intervals are maximized. Intervals being processed can be interrupted, but the value will be lost. This can also be viewed as a job scheduling problem where each job must be served immediately or else it is lost. This is a fundamental problem in scheduling and has been widely studied, and is also related to a number of online problems such as call control and bandwidth allocation (see e.g. [15, 4, 1]).

Related Work. For the basic problem where intervals are of the same length and with arbitrary weights, Woeginger [15] gave an optimal deterministic 4-competitive algorithm and a matching lower bound. In the paper, the open question of whether randomization can help give better algorithms was raised. Miyazawa and Erlebach [12] gave a 3-competitive randomized algorithm for the

special case where the weights of the intervals form a non-decreasing sequence. They also gave the first randomized lower bound of 1.25.

Many other variations exist for the problem. One is to allow variable interval lengths: Canetti and Irani [4] gave a randomized lower bound of $\Omega(\sqrt{\log \Delta / \log \log \Delta})$ and a randomized upper bound of $O(\log \Delta)$ where Δ is the ratio of the longest to shortest interval length. The deterministic case (with variable lengths) seems not studied before on its own. However, more general models have been studied. The problem of *online scheduling of broadcasts with restarts* is considered in [8, 5, 7, 16, 14]. In this broadcast scheduling problem, a set of pages is stored in a server, and requests for pages arrive online. Each request has a deadline by which the request has to be completed or else no profit is obtained. Different requests for the same page can be served together in one single broadcast. Requests being served can be interrupted but will need to restart from the beginning if it is to be broadcasted again. It can be seen that the interval scheduling problem is a special case of the broadcast scheduling problem (where all requests have tight deadlines and each request asks for a different page). In particular, there are matching upper and lower bounds $\Theta(\Delta / \log \Delta)$ [14, 16] for the deterministic case. These bounds also apply to the interval scheduling problem.

Another related problem, the *real-time job scheduling problem*, gives a bound on the interval scheduling problem w.r.t. a different parameter. In this problem, jobs have release times, deadlines and execution times. Preemption is allowed and preempted jobs can be resumed at the point where they were preempted. When all jobs have no laxity (i.e., execution time equals the difference between deadline and release time), this problem reduces to the interval scheduling problem. Matching upper and lower bounds of $(1 + \sqrt{k})^2$ are established in [2, 9], where k is the *importance ratio*, defined as the ratio of maximum to minimum weight of the jobs. (Here jobs have different lengths and the weight of a job is the value per unit length of the job.) Both bounds apply to the interval scheduling case.

Yet another case is where intervals can have different lengths and their weights are some function of their lengths. Seiden [13] gave a randomized 3.732-competitive algorithm for the case of *benevolent instances*, where (roughly speaking) the weight of an interval is either a convex or monotonic decreasing function of its length. If the weight of an interval is equal to its length, the nonpreemptive case was considered in [11]. They gave a randomized $O((\log \Delta)^{1+\epsilon})$ -competitive algorithm and a $\Omega(\log \Delta)$ lower bound.

Multiprocessor scheduling of intervals were studied in [6], giving an optimal (1-competitive) algorithm when all intervals have unit weight (and not necessarily equal length). Multiprocessor scheduling of jobs with different lengths and weights were studied in [10], with competitive ratio roughly $\Theta(\log k)$ where k is the importance ratio. If the profit is equal to the length, a tight bound of 2 is known for 2 processors [2].

Our Results. In this paper we consider the case where all intervals are of the same length. In fact our algorithm applies to the more general case where intervals may not have equal lengths but have *agreeable deadlines*, i.e. no interval being

strictly contained in another interval. This is because of the result that the class of interval graphs for agreeable-deadlines intervals is equal to the class of interval graphs for equal-length intervals (see e.g. [3]). In Section 3 we give a randomized online algorithm which is 3.618-competitive, which is lower than the optimal deterministic bound. The algorithm only uses a constant number of random bits as it only makes a single random choice when it starts. In Section 4 we consider lower bounds, giving an improved randomized lower bound of $4/3$ on the competitive ratio. For barely random algorithms that chooses between two algorithms of equal probabilities (which includes our proposed algorithm), we show a lower bound of 2.

There is a close relation between the randomized single-processor case and the deterministic multiprocessor case. We show in Section 5 that (with some modifications) our 3.618-competitive algorithm can be applied in the 2-processor case. The lower bounds also apply: we give a $4/3$ lower bound for the competitive ratio of any deterministic or randomized algorithms for any number of processors, and a deterministic 2 lower bound for 2 processors.

Due to space limitations some proofs are omitted and can be found in the full version of the paper.

2 Preliminaries

An interval I is specified by $s(I)$, its arrival time; $\ell(I)$, its length; and $|I|$, its weight. Since we only consider the case where all intervals are of the same length, we can, without loss of generality, assume $\ell(I) = 1$ for all I .

Intervals arrive online and the scheduling algorithm has to make decisions without knowledge of future intervals. When a scheduling algorithm completes an interval, it receives a *profit* equal to the weight of the interval. An interval being scheduled can be aborted (e.g. when an interval of larger weight arrives) but the value of the aborted interval will be lost. The objective of the algorithm is to maximize the total profit obtained by completing the intervals.

Let $A(S)$ denote the profit obtained by algorithm A on input instance S . An online algorithm A is *c-competitive* if for all input instances S , $OPT(S)/A(S) \leq c$ where OPT is the offline optimal algorithm which has knowledge of all future intervals and hence can schedule optimally. When A is a randomized algorithm, the definition of competitive ratio becomes $OPT(S)/E[A(S)] \leq c$ where the expectation is taken over the random choices of A . A randomized algorithm that only uses a constant number of random bits is called *barely random*.

3 A Randomized Algorithm

The Algorithm. Consider the simple deterministic algorithm Greedy_r: when an interval I arrives and the algorithm is currently executing another interval I' , it aborts I' and start I if $|I| \geq r|I'|$. If the machine is idle at that time then $|I'| = 0$, meaning that I will always get started. We call r the *abortion ratio*. This

algorithm is 4-competitive when $r = 2$ and is the best possible for deterministic algorithms [15].

Fix constants α, β, p where $1 < \alpha < \beta$ and $0 < p < 1$. The randomized algorithm **RGreedy** $_{\alpha, \beta, p}$ chooses to run one of the two deterministic algorithms **Greedy** $_{\alpha}$ and **Greedy** $_{\beta}$ with probability p and $1 - p$ respectively. It is barely random since the choice is only made in the beginning. Below we will analyze the competitive ratio of this algorithm.

Basic subschedules. Let A and B denote the schedules produced by **Greedy** $_{\alpha}$ and **Greedy** $_{\beta}$ respectively on a particular input instance. We define a *basic subschedule* to be a sequence of execution of intervals (I_1, \dots, I_k) , for $k \geq 1$, where I_i is aborted by I_{i+1} for $1 \leq i \leq k - 1$ and I_k is completed. That is, each basic subschedule consists of zero or more aborted intervals followed by a completed interval. Each of the two online schedules A and B can then be partitioned into a sequence of basic subschedules. In a basic subschedule (I_1, \dots, I_k) with abortion ratio r , we have $|I_i| \leq |I_{i+1}|/r$ because this is the condition for I_{i+1} to abort I_i .

Profit amortization. Consider a basic subschedule (I_1, \dots, I_k) with abortion ratio r . Only I_k is completed. Therefore the profit received for the whole basic subschedule is $|I_k|$. For the purpose of analysis, we will ‘amortize’ the profit of a basic subschedule to the individual intervals (not just the completed one) as follows. I_k transfers a profit of $|I_{k-1}|$ to I_{k-1} and keep the rest of $|I_k| - |I_{k-1}|$ profit to itself. Inductively, for $i = k - 1, \dots, 2$, I_i receives an amortized profit of $|I_i|$ from I_{i+1} ; it then transfers a profit of $|I_{i-1}|$ to I_{i-1} and keep the rest of $|I_i| - |I_{i-1}|$ to itself. For I_1 it keeps all its received profit $|I_1|$. Obviously, the total profit remains the same. From now on, unless explicitly stated otherwise, we will refer to the amortized profits.

Schedule segments. Consider a basic subschedule, (X_1, \dots, X_k) , of A . Let x_1, x_2, \dots, x_k be the weights of these intervals. Let t_i be the time when X_i is started, and define $t_{k+1} = t_k + 1$. During $[t_i, t_{i+1})$, OPT can start at most one interval, Z_i , with weight z_i . If there is no interval started by OPT during that time period, we simply skip this interval X_i from our consideration; thus X_i and t_i only refer to those intervals in A which have corresponding Z_i 's. This will only underestimate the profit of the online algorithm. By the property of basic subschedules we have $x_i \geq \alpha x_{i-1}$ for all $1 < i \leq k$. (Note that X_i and X_{i-1} may not be consecutive intervals in the basic subschedule because of the skipping just mentioned. Nevertheless the inequality remains true.)

Let u_i be the time when Z_i starts. At time u_i , **Greedy** $_{\alpha}$ must be serving some intervals with weight at least z_i/α or else **Greedy** $_{\alpha}$ would abort what it is serving and start Z_i instead. Thus $x_i > z_i/\alpha$ for all i . Similarly, at time u_i , the other algorithm **Greedy** $_{\beta}$ (if it is chosen instead) must be serving some interval, Y_i , of weight at least z_i/β , or else it will abort what it is serving and start Z_i instead. Denote by y_i the weight of this interval that **Greedy** $_{\beta}$ is serving at time u_i . We have $y_i > z_i/\beta$.

We make two observations about these y_i 's. First, any two Z_i 's must correspond to different Y_i 's. This is because each interval in OPT is completed and thus takes 1 unit of time, so $u_{i+1} \geq u_i + 1$ and hence Y_i and Y_{i+1} cannot be the same interval. Second, two consecutive Y_{i-1} and Y_i may or may not belong to the same basic subschedule in B . If they do, then we have $y_i \geq \beta y_{i-1}$. Note that even if they are in the same basic subschedule, they may not be consecutive intervals (there may be a number of abortions in-between), but even so the previous inequality remains true. If they are not in the same basic subschedule, we cannot say anything about y_i and y_{i+1} .

Therefore, we further split the basic subschedules in A and B into *segments* as follows. Let X_1 and Y_1 be the first interval in A and B respectively after the previous segment (initially they are simply the first intervals). A segment is then the set of intervals (X_1, \dots, X_n) from A and (Y_1, \dots, Y_n) from B such that at least one of X_n or Y_n is completed, and all other X_i and Y_i is aborted (directly or indirectly) by X_{i+1} and Y_{i+1} respectively. For a pair of corresponding segments, at least one of the two ending intervals is completed. In effect, intervals in a segment satisfy $x_i \leq x_{i+1}/\alpha$ (for those in A) and $y_i \leq y_{i+1}/\beta$ (for those in B).

Note that X_i 's and Y_i 's may not be consecutive intervals in a basic subschedule, as explained before. In the analysis we will ignore all those skipped abortions, e.g. in the profit amortization we treat Y_i and Y_{i+1} as if they are consecutive without giving any profit to any aborted intervals in-between, if any.

Bounding the expected profit. We now consider each such segment, where (X_1, \dots, X_n) is a segment from A , (Y_1, \dots, Y_n) is a segment from B , and (Z_1, \dots, Z_n) is the corresponding sequence of intervals in OPT .

The total profit of OPT in this segment is $\sum_{i=1}^n z_i$. As for the online algorithm, A has an amortized profit of at least $x_n - x_1/\alpha$ for this segment: the last interval has profit x_n and subsequently transferred to other intervals in this segment, except x_1 may transfer profit to a previously aborted interval, which has weight at most x_1/α . Similarly B receives an amortized profit of $y_n - y_1/\beta$ for this segment. The expected profit is thus at least $p(x_n - x_1/\alpha) + (1-p)(y_n - y_1/\beta)$. Note that the terms x_1/α and y_1/β would not be there if they are the first interval in a basic subschedule. But at least one of x_1 and y_1 must be such a first interval, since otherwise the segment would be extended to the front. Therefore, we can remove the smaller of these two terms from the expression. Thus the expected profit of the online algorithm is at least $px_n + (1-p)y_n - \max(px_1/\alpha, (1-p)y_1/\beta)$. We call the $\max(px_1/\alpha, (1-p)y_1/\beta)$ term the *amortized term*. The ratio R of optimal profit to the expected online profit in this segment is at most

$$\frac{\sum_{i=1}^n z_i}{px_n + (1-p)y_n - \max(px_1/\alpha, (1-p)y_1/\beta)} \quad (1)$$

We want to upper bound the above ratio, under the following constraints:

$$z_i < \min(\alpha x_i, \beta y_i), \quad x_i \leq x_{i+1}/\alpha, \quad y_i \leq y_{i+1}/\beta$$

Each interval served by OPT must belong to exactly one segment. Therefore, if we can upper bound the ratio of the total OPT profit to the expected online

profit, for all such segments, this gives a bound on the competitive ratio of the randomized algorithm.

For the rest of the paper, we fix $\alpha = \phi \approx 1.618$, $\beta = \phi^2 \approx 2.618$ and $p = 1/2$, where $\phi = (1 + \sqrt{5})/2$ is the golden ratio. We first state a technical lemma which will be required later.

Lemma 1. *Suppose $x_i = 1/\alpha^{n-i}$ and $y_i = y/\beta^{n-i}$ for all i . Then the function*

$$F(y) = \frac{\sum_{i=1}^n \min(\alpha x_i, \beta y_i)}{1 + y - 1/\alpha^n}$$

is increasing in y for $0 \leq y < \alpha/\beta$, and decreasing in y for $y > \alpha/\beta$.

Theorem 1. *The competitive ratio of $RGreedy_{\alpha,\beta,p}$ is $\phi + 2 \approx 3.618$ when $\alpha = \phi$, $\beta = \phi^2$ and $p = 1/2$.*

Proof. Consider a segment with (X_1, \dots, X_n) , (Y_1, \dots, Y_n) , and (Z_1, \dots, Z_n) . Without loss of generality, assume $x_n = 1$ and denote y_n simply as y . To maximize (1), observe that (for a fixed y) we should make x_i and y_i as large as possible, so that z_i 's are large and also x_1 and y_1 are large. This means $x_i = 1/\alpha^{n-i}$ and $y_i = y/\beta^{n-i}$. Together with $p = 1/2$, (1) becomes at most

$$\frac{2 \sum_{i=1}^n \min(\alpha x_i, \beta y_i)}{1 + y - \max(1/\alpha^n, y/\beta^n)} \quad (2)$$

We consider these cases:

Case 1: $y \leq (\beta/\alpha)^n$. In this case the amortized term is $1/\alpha^n$. The ratio (2) is equal to $2F(y)$ in Lemma 1, which we know is maximum when $y = \alpha/\beta$. At this value of y , all min terms in the numerator are βy_i terms and the ratio has maximum value

$$\begin{aligned} \frac{2\beta y(1 + 1/\beta + \dots + 1/\beta^{n-1})}{1 + y - 1/\alpha^n} &= \frac{2\beta y(1 - 1/\beta^n)/(1 - 1/\beta)}{1 + y - 1/\alpha^n} \\ &= \frac{\frac{2\alpha\beta}{\beta-1}(1 - 1/\beta^n)}{1 - 1/\alpha^n + \alpha/\beta} = \frac{2\phi^2(1 - 1/\phi^{2n})}{1 + 1/\phi - 1/\phi^n}. \end{aligned}$$

This is at most $\phi + 2 \approx 3.618$ for any value of n (maximum occurs when $n = 2$).

Case 2: $y > (\beta/\alpha)^n$. In this case all min terms are the αx_i terms and the amortizing term is y/β^n . So the ratio is

$$\begin{aligned} \frac{2(\alpha + 1 + \dots + 1/\alpha^{n-2})}{1 + y - y/\beta^n} &= \frac{2\alpha(1 - 1/\alpha^n)/(1 - 1/\alpha)}{1 + y(1 - 1/\beta^n)} \\ &\leq \frac{2\alpha(1 - 1/\alpha^n)/(1 - 1/\alpha)}{1 + (\beta/\alpha)^n - 1/\alpha^n} = \frac{2\phi^3(1 - 1/\phi^n)}{1 + \phi^n - 1/\phi^n}. \end{aligned}$$

This is at most ϕ^2 for any value of n .

Therefore in either case the ratio is at most $\phi + 2$. \square

We can show that there is an instance which actually attains the competitive ratio of 3.618 using our algorithm (with these chosen parameters), so that the analysis is tight.

4 Lower Bounds

4.1 Randomized algorithms

Theorem 2. *No randomized algorithm for interval scheduling has competitive ratio better than $4/3$.*

Proof. We will use Yao's principle which states that the randomized lower bound can be obtained by bounding $E[OPT]/E[A]$ for any deterministic algorithm A over a probability distribution of input instances. (See for example, [12].) Thus we define an input distribution as follows. Let (r, w) denote the release time and weight respectively of an interval. Fix a large even integer n . Define $n + 1$ intervals, I_0, I_1, \dots, I_n , such that for $0 \leq i \leq n - 1$, $I_i = (i/2, v_i)$ where $v_i = 2^i$, and $I_n = (n/2, v_n)$ where $v_n = 2^{n-1}$. Define n sets of intervals, S_1, S_2, \dots, S_n , such that $S_i = \{I_0, I_1, \dots, I_i\}$ for $1 \leq i \leq n$. Finally, we define our distribution of inputs to be one such that S_i occurs with probability $p_i = 1/2^i$ for $1 \leq i \leq n - 1$ and S_n occurs with probability $p_n = 1/2^{n-1}$.

Since any I_i does not overlap with I_{i+2} , we have $OPT(S_i) = 1 + 4 + \dots + 2^i = \frac{4^{i/2+1}-1}{3}$ if i is even, and $OPT(S_i) = 2 + 8 + \dots + 2^i = \frac{2(4^{(i+1)/2}-1)}{3}$ if i is odd, and $OPT(S_n) = 1 + 4 + \dots + 2^{n-2} + 2^{n-1} = \frac{4^{n/2}-1}{3} + 2^{n-1}$. Hence

$$\begin{aligned} E[OPT] &= \sum_{i=2, i \text{ even}}^{n-2} \frac{4^{i/2+1}-1}{3 \cdot 2^i} + \sum_{i=1, i \text{ odd}}^{n-1} \frac{2(4^{(i+1)/2}-1)}{3 \cdot 2^i} + \frac{4^{n/2}-1}{3 \cdot 2^{n-1}} + 1 \\ &= 4n/3 - o(n). \end{aligned}$$

We now derive an upper bound on the expected profit of an arbitrary deterministic algorithm A on our input distribution. More specifically, for $i = 1, \dots, n$, we let Q_i be the contribution to the expected profit of A on I_{i-1}, \dots, I_n when the input is one of S_i, \dots, S_n .

Consider the case when the input is S_n . This happens with probability p_n . When I_n arrives at time $n/2$, A may or may not be serving another interval. If it does, it must be serving I_{n-1} . Since we choose $v_{n-1} = v_n$, A will obtain at most a profit of v_n whether it aborts I_{n-1} or not. Thus, $Q_n \leq p_n v_n$.

Now, suppose the input is either S_{n-1} or S_n . When I_{n-1} arrives at time $(n-1)/2$, A may or may not be serving I_{n-2} . There are two cases.

Case 1: A is serving I_{n-2} and it continues until its completion. Then A gains a profit of v_{n-2} on I_{n-2} whether the input is S_{n-1} or S_n . Further, it can gain an expected profit of at most Q_n on I_{n-1} and I_n when the input is S_n . Hence, $Q_{n-1} \leq (p_{n-1} + p_n)v_{n-2} + Q_n$.

Case 2: A is not serving I_{n-2} or if it aborts I_{n-2} . Then A may have an expected profit of $p_{n-1}v_{n-1}$ on I_{n-1} when the input is S_{n-1} and an expected profit of Q_n on I_{n-1} and I_n when the input is S_n . Note that the input being S_{n-1} and being S_n are two disjoint events. Thus, $Q_{n-1} \leq p_{n-1}v_{n-1} + Q_n$.

Setting $(p_{n-1} + p_n)v_{n-2} + Q_n = p_{n-1}v_{n-1} + Q_n$ (which is satisfied by requiring $v_{n-2} = \frac{p_{n-1}}{p_{n-1} + p_n}v_{n-1}$), we have $Q_{n-1} \leq p_{n-1}v_{n-1} + Q_n$ no matter what A does.

In general, consider the case when the input is one of S_i, \dots, S_n . When I_i arrives at time $i/2$, A may or may not be serving I_{i-1} and we consider the following cases.

Case 1: A is serving I_{i-1} and it continues with it until completion. Then A gains an expected profit of $(p_i + \dots + p_n)v_{i-1}$ on I_{i-1} (no matter what the true input is) and an expected profit of Q_{i+1} on I_i, \dots, I_n when the input is S_{i+1}, \dots, S_n . Thus, $Q_i \leq (p_i + \dots + p_n)v_{i-1} + Q_{i+1}$.

Case 2: A is not serving I_{i-1} or if it aborts I_{i-1} . Then A gains an expected profit of $p_i v_i$ on I_i when the input is S_i , and an expected profit of Q_{i+1} on I_i, \dots, I_n when the input is one of S_{i+1}, \dots, S_n . Hence $Q_i \leq p_i v_i + Q_{i+1}$.

Setting $v_{i-1} = \frac{p_i}{p_i + \dots + p_n} v_i$, we have $(p_i + \dots + p_n)v_{i-1} + Q_{i+1} = p_i v_i + Q_{i+1}$. So $Q_i \leq p_i v_i + Q_{i+1}$.

One can easily check that setting p_i and v_i as mentioned earlier, the conditions $v_{i-1} = \frac{p_i}{p_i + \dots + p_n} v_i$ for $1 \leq i \leq n$, are satisfied and the total expected profit of A is $Q_1 \leq p_1 v_1 + \dots + p_n v_n = n$.

Therefore, $E[OPT]/E[A] \rightarrow 4/3$ for $n \rightarrow \infty$. \square

Remarks on benevolent instances. The lower bound construction does not rely on the exact lengths of the intervals. The only requirement on the lengths is that I_i and I_{i+1} intersect while I_i and I_{i+2} do not. Therefore, the lower bound also holds for benevolent instances; we just create the instances with the specified weights and adjust the lengths accordingly.

4.2 Barely random algorithms

Our randomized algorithm in Section 3 chooses between two deterministic algorithms with equal probabilities. We next show a lower bound on such algorithms.

Theorem 3. *No barely random algorithms that choose between two deterministic algorithms with equal probabilities can be better than 2-competitive.*

Proof. Suppose on the contrary there exists such a randomized algorithm which is $(2 - \epsilon)$ -competitive for some constant $\epsilon > 0$. Let D1 and D2 be the two deterministic algorithms. We construct an adversarial request sequence to show that this results in a contradiction.

Consider a set of a large number of intervals where each interval differs from the previous one by arriving slightly later and having a slightly larger weight (difference in weight being δ). The minimum weight of intervals in this set is 1 and the maximum weight is α . Here δ is a sufficiently small and α a sufficiently large constant to be chosen later. The last interval arrives before the deadline of the first interval, and hence any algorithm can serve at most one interval in this set. (This is the set of intervals used in [15].) Given this set of intervals, let x and y be the weights of intervals chosen by D1 and D2, where without loss of generality, assume $x \leq y$. We emphasize that the adversary knows the values of x and y . We consider the following cases.

Case 1: $x = y = 1$. Both D1 and D2 obtains a profit of 1 while OPT schedules the heaviest interval giving a profit of α . So the competitive ratio is α .

Case 2: $x = y \neq 1$. One more interval of weight y is released just before the deadline of the y in the set. Both D1 and D2 either continue with the x or y , or abort and switch to the new y . In either case their profit is at most y . The adversary schedules the interval in the set just before y , together with the new y , giving a profit of $(y - \delta) + y$. Hence the competitive ratio is $2 - \delta/y > 2 - \delta$.

Case 3: $1 = x < y$. D1 and D2 gets a profit of 1 and y respectively while OPT gets α . Thus the competitive ratio is $\alpha/((1+y)/2) \geq 2\alpha/(1+\alpha) = 2 - 2/(1+\alpha)$.

Case 4: $1 < x < y$. The adversary releases another interval with weight y just before the deadline of x in the set. We distinguish two subcases.

If D1 does not abort x in favour of the new y , no more intervals are released. (We remark that the adversary knows the response of D1 and can make requests accordingly.) In this case D1 and D2 get a profit of x and y respectively, while OPT gets $x - \delta + y$. Then the competitive ratio $= (x + y - \delta)/((x + y)/2) = 2 - 2\delta/(x + y) > 2 - 2\delta/(1 + 1) = 2 - \delta$.

If D1 aborts x and serves y , then one more interval of weight y arrives just before the deadline of y in the original set. Then both D1 and D2 gets a profit of y no matter what they do, and OPT gets a profit of $y - \delta + y$. The competitive ratio is $(2y - \delta)/y = 2 - \delta/y > 2 - \delta$.

Considering all cases, the competitive ratio is at least $\min\{\alpha, 2 - \delta, 2 - 2/(1 + \alpha)\}$. By choosing $\delta < \epsilon$ and $\alpha > \max(2 - \epsilon, 2/\epsilon - 1) = 2/\epsilon - 1$, we have the competitive ratio being at least $2 - \epsilon$. \square

5 The Multiprocessor Case

In this section we consider the case of using more than one processor to schedule the intervals. We will see that the cases of randomization and multiple processors are closely related. We first show that the idea of the barely random algorithm in Section 3 can be used to give a deterministic 2-processor algorithm with the same competitive ratio. Then we show that the lower bounds in Section 4 can also be carried to the multiprocessor case; namely, that no deterministic or randomized algorithm can be better than $4/3$ -competitive for any number of processors m , and no 2-processor deterministic algorithm can be better than 2-competitive.

5.1 A 2-processor Algorithm

We consider the following deterministic 2-processor algorithm. Call the two processors P1 and P2. In simple terms, P1 runs Greedy $_{\alpha}$ whereas P2 runs Greedy $_{\beta}$. Specifically, suppose the two processors are running intervals I_1 and I_2 respectively. When a new interval I arrives, if $|I| < \alpha|I_1|$ and $|I| < \beta|I_2|$ then I is rejected. If one of $|I| \geq \alpha|I_1|$ and $|I| \geq \beta|I_2|$ is true, the corresponding I_1 or I_2 is aborted and I is started on that processor. If I is at least as large as both $\alpha|I_1|$ and $\beta|I_2|$, it aborts I_2 and start I on P2. (This is the only difference from the randomization case: since the two processors cannot be doing the same interval, we need some way of tie-breaking.) A processor which has completed its interval will become idle. Note that an idle processor is regarded as executing a weight-0

interval. Therefore if P1 is idle and $|I| \geq \beta|I_2|$, it will still abort I_2 (and P1 remains idle). Again we set $\alpha = \phi$ and $\beta = \phi^2$.

We will separately bound the value of the two optimal offline schedules produced by the two processors, OPT1 and OPT2. As before, we divide the schedule into basic subschedules and segments. With the same notation as in Section 3, consider a segment where OPT1 schedules (Z_1, \dots, Z_n) , P1 schedules (X_1, \dots, X_n) , and P2 schedules (Y_1, \dots, Y_n) . We will show the same bound on the competitive ratio, i.e. $2 \sum z_i / (x_n + y_n - \max(x_1/\alpha, y_1/\beta)) \leq \phi + 2$. Therefore over the whole OPT1, $OPT1 / ((P1 + P2)/2) \leq \phi + 2$ where $P1$ and $P2$ are the two schedules. Here $OPT1$, $P1$ and $P2$ represent both the schedules and their profits. Since OPT2 can be analyzed similarly, we have $OPT2 / ((P1 + P2)/2) \leq \phi + 2$. Adding these two together, $OPT1 + OPT2 \leq (\phi + 2)(P1 + P2)$ and therefore the algorithm is $(\phi + 2)$ -competitive. In the analysis below we only consider OPT1.

We call (z_k, x_k, y_k) in a segment a *triplet*. We first make an observation:

Lemma 2. *For any triplet (z_k, x_k, y_k) , one of the following two cases holds: (i) $x_i > z_i/\alpha$ and $y_i > z_i/\beta$, (ii) $z_i = y_i$ and $x_i \leq z_i/\alpha$.*

We call triplets of case (i) *normal triplets* and those of case (ii) *violating triplets*. The main idea of the competitiveness proof is as follows: if there are no violating triplets in a segment, then we are done by the same proof as in the randomized algorithm. If there are violating triplets, we further divide the segment into *subsegments* so that each subsegment has at most one violating triplet at the beginning of the subsegment. We then perform a similar analysis to the randomized algorithm on each subsegment. There is a small difference in the amortized terms: both the x_1/α and y_1/β terms may be subtracted, since the last pair of intervals in the previous subsegment may not be completed, and hence do not have any real profit. So the amortized term may sometimes become $x_1/\alpha + y_1/\beta$ instead of $\max(x_1/\alpha, y_1/\beta)$. We omit the proof to this theorem:

Theorem 4. *The algorithm is $\phi + 2 \approx 3.618$ -competitive for 2 processors.*

5.2 Lower Bounds

The proofs of the following theorems use almost identical constructions to that in Theorems 2 and 3, so we omit the proofs.

Theorem 5. *No deterministic or randomized algorithm for online interval scheduling on m processors is better than $4/3$ -competitive, for any m .*

Theorem 6. *No deterministic algorithm for online interval scheduling on 2 processors is better than 2-competitive.*

6 Conclusion

In this paper we give the first randomized algorithm and improved lower bounds for the online interval scheduling problem. The gap between the upper and lower

bounds remains wide, however. It may be possible to generalize the barely random algorithm to use 3 or more deterministic algorithms but we encounter some technical difficulties in extending the technique here. Algorithms for three or more processors will also yield randomized algorithms for the single processor case.

References

1. B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosen. Competitive non-preemptive call control. In *Proc. 5th SODA*, pages 312–320, 1994.
2. S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang. On the competitiveness of on-line real-time task scheduling. *Real-Time Systems*, 4:125–144, 1992.
3. K. P. Bogart and D. B. West. A short proof that proper = unit. *Discrete Mathematics*, 201:21–23, 1999.
4. R. Canetti and S. Irani. Bounding the power of preemption in randomized scheduling. *SIAM Journal on Computing*, 27(4):993–1015, 1998.
5. W.-T. Chan, T.-W. Lam, H.-F. Ting, and P. W. H. Wong. New results on on-demand broadcasting with deadline via job scheduling with cancellation. In *Proc. 10th COCOON*, LNCS 3106, pages 210–218, 2004.
6. U. Faigle and W. M. Nawijn. Greedy k -coverings of interval orders. Technical Report 979, University of Twente, 1991.
7. S. P. Y. Fung, F. Y. L. Chin, and C. K. Poon. Laxity helps in broadcast scheduling. In *Proc. 9th Italian Conference on Theoretical Computer Science*, LNCS 3701, pages 251–264, 2005.
8. J.-H. Kim and K.-Y. Chwa. Scheduling broadcasts with deadlines. *Theoretical Computer Science*, 325(3):479–488, 2004.
9. G. Koren and D. Shasha. D^{over} : An optimal on-line scheduling algorithm for overloaded uniprocessor real-time systems. *SIAM Journal on Computing*, 24:318–339, 1995.
10. G. Koren and D. Shasha. MOCA: A multiprocessor on-line competitive algorithm for real-time system scheduling. *Theoretical Computer Science*, 128(1-2):75–97, 1994.
11. R. J. Lipton and A. Tomkins. Online interval scheduling. In *Proc. 5th SODA*, pages 302–311, 1994.
12. H. Miyazawa and T. Erlebach. An improved randomized on-line algorithm for a weighted interval selection problem. *Journal of Scheduling*, 7(4):293–311, 2004.
13. S. S. Seiden. Randomized online interval scheduling. *Operations Research Letters*, 22(4–5):171–177, 1998.
14. H.-F. Ting. A near optimal scheduler for on-demand data broadcasts. In *Proc. 6th Italian Conference on Algorithms and Complexity*, LNCS 3998, pages 163–174, 2006.
15. G. J. Woeginger. On-line scheduling of jobs with fixed start and end times. *Theoretical Computer Science*, 130(1):5–16, 1994.
16. F. Zheng, S. P. Y. Fung, W.-T. Chan, F. Y. L. Chin, C. K. Poon, and P. W. H. Wong. Improved on-line broadcast scheduling with deadlines. In *Proc. 12th COCOON*, LNCS 4112, pages 320–329, 2006.