

CSC2421: Online and Other Myopic Algorithms

Fall 2026

Allan Borodin

October 19, 2026

Lecture 10

Announcements

- Please submit final specification of your individual and group projects.
- As I announced on Quercus, in the real-time model with revoking, the Fung 2014 paper that is posted on the course web page does achieve a 4-competitive ratio for the Throughput Maximization problem with proportional weights. Michael He shows how to extend that result to C-Benevolent weights, and single processing time instances with arbitrary weights. We expect that a similar proof also obtains a 4-competitive ratio for D-competitive weight functions.

I have posted a preliminary draft of the C-Benevolent result on Quercus.

- We basically use the analysis in Volume 2, Chapter 5 that we have for our 5-competitive algorithm with restarts. I had initially thought that we could not obtain a constant ratio with just revoking. Now we know that a 4-competitive ratio is possible using just revoking. (A 4-competitive ratio is the best we can obtain for any C-Benevolent weight function even for intervals by a result in Woeginger.)

Today's Agenda

- A short discussion about theory vs real world applications. This wasn't planned but I was asked about the value of worst case results especially when they seem inconsistent with applications.
- Finish up sketch of the 2-competitive result for Unweighted Online Interval Selection in the ROA model.
- The algorithm and a sketch of competitive analysis for the Throughput Maximization problem with C-Benevolent weights.
- Some additional results in the paper with Michael He and Nadim Mottu.
- Simulating online randomized algorithms with adversarial sequences by a deterministic algorithm with random order arrivals (the ROA model)

Unweighted interval selection in the ROA model

Surprisingly, the same simple algorithm (as used for the adversarial sequence $2k$ ratio) has competitive ratio 2 in the random order arrival model. We conjecture that this is the optimal ratio. So far the best deterministic and randomized lower bound is $\frac{4}{3} - \epsilon$ (for any $\epsilon > 0$) in the ROA model. The lower bound is due solely to Claude.

The analysis for both the upper and lower bounds is much more substantial. It is also interesting that the same charging function can be used but, as we said, the analysis is quite different. We will want to show that $\frac{OPT}{\mathbb{E}[ALG]} \leq 2$ for all input sequences, noting that now ALG is a random variable depending on the ROA permutation whereas OPT is not effected by the random order of the sequence.

The history of the result.

The paper was rejected from a conference with one referee having suspicious about our cludgy argument. With David Zhang, we asked Claude to verify our proof and Claude showed that our argument was not correct. Claude then provided a different analysis with the improved ratio.

The high level idea of the analysis for online interval selection in the ROA arrival

Let $\mathcal{I}$ denote the set intervals in the input sequence. We now have a charging function $f_\pi : OPT(\mathcal{I}) \rightarrow ALG_\pi(\mathcal{I})$ where π is the permutation applied to the intervals in $\mathcal{I}$.

At any point in the execution of the algorithm we have $\Phi_\pi(I) = |I' \in OPT : f_\pi(I') = I|$. Therefore $OPT = \sum_{I \in ALG} \Phi_\pi(I)$. But now we are going to account for charges indirectly.

Let's consider the predecessor chain of revoking. Let $I \in ALG$. The predecessor chain is the maximal list of intervals $(P_1, P_2, \dots, P_k = I)$ such that P_i was temporarily accepted by ALG and then later replaced by P_{i+1} when P_{i+1} is contained in P_i .

Woeginger defined predecessor chains to analyze C-Benevolent, D-Benevolent and single interval length adversarial instances of interval selection in the real time model with revoking. With Michael He and Nadim Motu, we the analysis to the Throughput problem.

Continuing the high level idea of the ROM result for unweighted interval selection.

If $(P_1, P_2, \dots, P_k = I)$ is a predecessor chain, then since this chain is maximal, P_1 was (temporarily) accepted and didn't conflict with a previous *ALG* interval. We will call such an interval a *root*. It can be that $k = 1$ in which case I was not revoked.

Abusing notation, if I_r is a root, let $\Phi_r = \Phi(I_r)$ and $\Phi_r = 0$ if I_r is not a root. We will sometimes just refer to I_r as r .

It is not hard to see that $ALG = \sum_{r \text{ is a root}} \mathbf{1}$ and $OPT = \sum_{I \in ALG} \Phi(I) = \sum_{r \text{ is a root}} \Phi_r$. Note that $[\mathbf{1} | r \text{ is a root}]$ and Φ_r are random variables in terms of the sequence permutation π .

Lemma

$$\frac{OPT}{\mathbb{E}[ALG]} \leq \max_r \mathbb{E}[\Phi_r | r \text{ is a root}].$$

The Lemma and why this indirect charging seems essential

$$\frac{OPT}{\mathbb{E}[ALG]} \leq \max_r \mathbb{E}[\Phi_r | r \text{ is a root}].$$

Proof

$$\begin{aligned} OPT &= \sum_r \text{Prob}[r \text{ is a root}] \cdot \mathbb{E}[\Phi_r | r \text{ is a root}] \leq \\ &\sum_r [\text{Prob}[r \text{ is a root}] \cdot \max_r \mathbb{E}[\Phi_r | r \text{ is a root}]] \text{ and} \\ &\sum_r \text{Prob}[r \text{ is a root}] = \mathbb{E}[ALG]. \end{aligned}$$

Note: The event that r is a root depends only on the arrivals up to and including r ; that is, the event only depends on the past.

In our original proof we had

$$\frac{OPT}{\mathbb{E}[ALG]} \leq \max_I \mathbb{E}[\Phi(I) | I \in ALG].$$

The condition that $I \in ALG$ depends on the future which makes the analysis difficult and our analysis faulty.

The Algorithm for Throughput Maximization with C-Benevolent Weights

I will now switch back to our use of fractional constant competitive ratios for maximization problems.

Let $x_i = d_i - p_i$, be the “expiry time” or “latest start time” of job J_i ; that is, the time after which the job would not be able to meet its deadline even if it is not interrupted. Alternatively, if $s_i = d_i - (r_i + p_i)$ is the slack of the job, $x_i = r_i + s_i$. A job is pending if it has arrived and can still be completed by its deadline. Let $w(J)$ be the weight of job J .

The Algorithm is based on the idea of not interrupting a job J_{curr} unless there is a pending job J_j that has hit it's expiry time x_j and $w(J_j) > 2w(J_{curr})$.

When jobs arrive at some time T , we place these new jobs on the queue of pending jobs which is sorted according to non-decreasing expiry times. In the $\frac{1}{5}$ competitive ratio result we have in the text, we were interrupting and scheduling jobs as early as we could which in hindsight makes more sense when you have resumming.

The pseudo code for the algorithm we just described

I won't post the algorithm in the slides since it is posted on Quercus and I prefer not to post unpublished results and I think we can improve things.

Algorithm 1 Maximum-Weight Scheduling with Latest-Start Revoking

```
1:  $J_{\text{curr}} \leftarrow \text{None}$  ▷ Job currently being processed
2:  $Q \leftarrow \emptyset$  ▷ Pending jobs
3:  $\mathcal{A} \leftarrow \emptyset$  ▷ Completed jobs
4:  $T \leftarrow 0$  ▷ Current time

5: while there are future arrivals,  $J_{\text{curr}} \neq \text{None}$ , or  $Q \neq \emptyset$  do
6:   Advance  $T$  to the next release, completion, or future latest start in  $Q$ 
7:   if  $J_{\text{curr}}$  completes at time  $T$  then
8:      $\mathcal{A} \leftarrow \mathcal{A} \cup \{J_{\text{curr}}\}$ 
9:      $J_{\text{curr}} \leftarrow \text{None}$ 
10:  Remove from  $Q$  all jobs  $J_j$  with  $T > x_j$  ▷ Expired
11:  Insert into  $Q$  all feasible jobs released at time  $T$ 

12:  if  $J_{\text{curr}} \neq \text{None}$  and  $Q \neq \emptyset$  then
13:     $M \leftarrow \max_{J \in Q} w(J)$ 
14:    if there is a job  $J_j \in Q$  with  $x_j = T$ ,  $w(J_j) = M$ , and  $w(J_j) > 2w(J_{\text{curr}})$  then
15:      Revoke  $J_{\text{curr}}$  permanently
16:      Choose such a job  $J_j$  and remove it from  $Q$ 
17:      Start  $J_j$ ;  $J_{\text{curr}} \leftarrow J_j$ 

18:  if  $J_{\text{curr}} = \text{None}$  and  $Q \neq \emptyset$  then
19:    Choose a maximum-weight job  $J_j \in Q$  and remove it from  $Q$ 
20:    Start  $J_j$ ;  $J_{\text{curr}} \leftarrow J_j$ 
```

End of Monday, October 19 class

Today, I presented the current algorithm for C-Benovolent weighted instances of the Throughput Maximization problem. I was asked to clarify the problem. We are assuming that for every job j , the weight $w(j)$ of a job is $f(p(j))$ where $p(j)$ is the processing time of the job and f is a specific C-Benvolent function, meaning that f is strictly increasing and convex.

Next class we will sketch the proof and go on to discuss some other active research I am involved in.