

# **CSC2421: Online and Other Myopic Algorithms**

## **Fall 2026**

Allan Borodin

October 21, 2026

# Lecture 11

## Announcements

- Next week is reading week (no class) and I expect to have everyone's final individual and group projects sent to me. Moreover, I hope there will be significant progress on your projects during reading week.
- My upcoming schedule. I am attending the FOCS conference which takes place at NYU in NYC, Nov 8-11. I will have to cancel the Monday Nov 9 class but will return late Tuesday, Nov 10 so we will have a class on Wednesday, Nov 11.
- I also have a family occasion, which will make me miss the Wednesday, November 25 class.
- That leaves us with 10 more classes, not counting "make-up" Wednesday December 9. As mentioned, we have 12 individual and 5 group projects. If we allow 25 minutes per talk, then we only have 2 talks per class. So we need 8 and a half classes.
- My guess is that we may likely have to use make-up Wednesday to be able to have all 17 talks. The sooner some individual or group can present, the better. **So I need volunteers for early presentations.**

# Today's Agenda

I will use fractional constant competitive ratios for maximization problems today although Fung and others use ratios greater than 1. .

- The Fung [2014] algorithm for proportionally weighted Throughput with ratio  $1/4$  in the real-time with revoking model.
- A fallacious argument for converting an online algorithm for interval selection problems to real-time algorithm for throughput maximization problems.

An example where the Fung algorithm will be worse than a ratio  $1/4$  for C-Benevolent weights.

- Michael He's algorithm for obtaining a  $1/4$  ratio for C-Benevolent weights.
- A  $1/5$  ratio for D-Benevolent weighted throughput maximization. Modifying the Wioeginger real-time algorithm for interval selection.
- Other results in the paper with Michael and Nadim Mottu.

# The Fung [2014] algorithm for proportionally weighted Throughput

This is almost (except for the tie breaking rule which is essential for Fung's algorithm) the algorithm that I was (incorrectly) proposing would “work” for C-Benevolent weighted instances. It seems more natural to me.

## Algorithm 1 in Fung for proportionally weighted Throughput maximization:

- Upon arrivals at any time  $T$ , add the newly arrived jobs to the queue of pending jobs.
- If the machine is idle (i.e., the last scheduled job has completed), schedule a job with the earliest expiry date  $x_i = d_i - p_i$ .
- Fung doesn't say this but we need to break ties by selecting the job with the maximum weight.
- Let  $E_T = \{J_i : x_i = T\}$ . If a job  $J_{curr}$  is still executing at time  $T$  and  $E_T \neq \emptyset$ , then interrupt  $J_{curr}$  and revoke  $J_{curr}$  iff  $\max_{J_i \in E_T} w(J_i) > 2w(J_{curr})$ ; and schedule any max weight job in  $E_T$ .

## A Falacious Argument for how to derive Fung's algorithm

I thought that Fung's algorithm is simply ignoring the release times and scheduling by expiry times which is then like scheduling intervals. Such an algorithm *ALG* would then complete jobs with total weight  $W$  if there is as an online (not real-time) interval selection algorithm  $A'$  that obtains weight  $W$ .

This is true but there is a fallacy in claiming that  $A$  inherits the same competitive ratio. **What is the fallacy?**

# A Falacious Argument for how to derive Fung's algorithm

I thought that Fung's algorithm is simply ignoring the release times and scheduling by expiry times which is then like scheduling intervals. Such an algorithm  $ALG$  would then complete jobs with total weight  $W$  if there is as an online (not real-time) interval selection algorithm  $A'$  that obtains weight  $W$ .

This is true but there is a fallacy in claiming that  $A$  inherits the same competitive ratio. **What is the fallacy?**

OPT can possibly obtain a better weight for completed jobs than by what OPT can obtain using only the expiry date and selecting jobs as intervals. It can do so since it knows about jobs when they arrive and jobs can have slack  $s_i = d_i - (r_i + p_i)$  which OPT can utilize.

This is similar to the (at first, perhaps surprising) fact that with resumming there is no constant competitive deterministic algorithm for the throughput problem with resumming, while there is a  $\frac{1}{2}$  competitive algorithm for this problem for the restarting problem.

## An example by Michael showing that the Fung algorithm *ALG 1* has a competitive ratio worse than $1/4$ for C-Benevolent weighted throughput

Let the weight function be  $w(J_i) = p(J_i)^2$ . Let job  $J_i = (r_i, p_i, s_i, w_i)$ , where  $s_i$  is the slack. Note that the deadline  $d_i = (r_i + p_i + s_i)$  and also  $w_i$  do not have to be stated.

Here is an example showing that  $OPT > 4 \cdot ALG\ 1$ .

$J_1 = (0, 6, 0)$ ,  $J_2 = (5, 9, 0)$ ,  $J_3 = (13, 13, 0)$ ,  $J_4 = (1, 24, 324)$ ,  $J_5 = (1, 24, 324)$ .

Thus we have three jobs with no slack (i.e. intervals) and two identical jobs.

# Comparing OPT with Fung's ALG 1

By Fung's algorithm:

At time 0,  $J_1$  is started.

At time 1,  $J_4$  and  $J_5$  arrive, but the algorithm ignores them because they are not expiring.

At time 5, tight job  $J_2$  arrives,  $J_1$  is revoked and  $J_2$  is started i sin ce  $81 > 2 * 36$

At time time 13, job  $J_3$  arrives,  $J_2$  is revoked and  $J_3$  is started since  $169 > 2 * 81$ . At time 25, both  $J_4$  and  $J_5$  reach their latest start time  $(1+24)$ , but since  $324 < 2 < 169$ , the algorithm does not start either of them. At time 26,  $J_3$  comoletes, but both  $J_4$  and  $J_5$  have expired.

So ALG=169, while OPT completes  $J_1$  in  $[0, 6)$ , completes  $J_4$  in  $[6, 24)$ , iand completes  $J_5$  in  $[24, 42)$ , resulting

$$OPT = 36 + 324 + 324 = 684 > 676 = 4 * 169 = 4 * ALG 1.$$

The significant difference between proportional weights and an arbitrary C-Benevolent function  $w(J)$  is that for proportional weightes  $w(J) > w(J') \implies p(J) > p(J')$ .

# The New Algorithm for Throughput Maximization with C-Benevolent Weights

Again 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. And again 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  $x_i$ ,  $w(J_j) > 2w(J_{curr})$  and  $J_j$  has maximum weight amongst *all* pending jobs.

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.

As in the  $\frac{1}{5}$  ratio result, we only interrupt (and now revoke) a currently running if  $w(J_j) > 2w(J_{curr})$ .

## The pseudo code for the algorithm we just described

The unpublished not with the pseud code and a proof of the competitive ratio is posted on Quercuse. I prefer not to post unpublished results and I also think we can improve the exposition. This result will be included along with other results in a forthcoming paper by Michael He, Nadim Mottu and myself. 1

## Some comments on the algorithm

I haven't read the proof for the Fung [2014] proportionally weighted case but I assume it is implicitly using the predecessor chain concept in Woeginger [1994]. Fung does cite the Woeginger paper.

The idea of revoking  $J_{curr}$  by some  $J_i$  iff  $w(J_i) > 2w(J_{curr})$  occurs in the following papers (and probably other papers):

- 1 The  $1/4$  competitive ratio in Woeginger for interval selection (with C-Benevolent, D-Benevolent and single processing times weighted instance) in the real time model with revoking.
- 2 It also occurs in Garay et al [1997] for the online (with revoking) result obtaining the ratio  $2 + \sqrt{5}$ . for call processing (interval selection) problem. This paper does not reference Woeginger.
- 3 It occurs in the Erlebach and Spieksma [2003] priority algorithm (with revoking) for interval election and JISP problems for proportional and arbitrary weightss. It is also in Horn's MSc thesis.

## Continuation of the comments on the algorithm

**Comment:** I didn't think a deterministic constant competitive ratio was possible if we only allow revoking and not restarting for proportional weights. But we knew that for online interval selection, while there is no constant ratio for unweighted interval selection in the online model with revoking there is the Garay et al  $2 + \sqrt{5}$  ratio for proportional weights. We have the same juxtaposition of results in the paper we are writing.

**Theorem:** For the unweighted throughput problem, when there are at most  $k$  distinct processing times, there is a deterministic real time algorithm (with revoking) that achieves a  $\frac{1}{2k}$  ratio and there is no such algorithm that achieves a ratio better than  $\frac{1}{k}$ . So there is a gap. We conjecture the correct ratio is  $\frac{1}{2k}$

## High Level Idea of the Competitive Analysis for the C-Benevolent weighted Throughput algorithm.

We follow Woeginger's predecessor and charging. A predecessor chain is a sequence of jobs  $J_1, J_2, \dots, J_s$  where  $J_{i+1}$  caused the algorithm to revoke  $J_i$  and  $J_s$  is accepted by the algorithm.

The predecessor chains are disjoint (although we can have a chain  $C$  followed by a chain  $C'$  where  $C'$  starts where  $C$  ends). Therefore we can basically restrict our analysis to a specific chain.

Our goal is to charge the intervals in OPT that intersect a chain to that chain. An OPT interval can only intersect two chains. When it intersects two chains, that may cause us to merge chains.

Let  $p_1, \dots, p_s$  be the processing times for the algorithm in a chain  $C$  and let  $q_1, \dots, q_t$  be the processing times for OPT intervals intersecting the chain. We can conceptually merge OPT intervals and since the weight function is increasing and convex this can only help OPT. We can then assume  $t \leq s$  and add fictional OPT intervals with 0 processing times so that  $t = s$ .

## Finishing the sketch of the analysis

We will want to show that  $\sum_i^k p_i \geq \sum_i^k q_i$  for all  $k \leq s$ .

We can then use Karamata's Inequality to show that  $\sum_i^s w(p_i) \geq \sum_i^s w(q_i)$ .

We will charge OPT intervals  $J$  to Type A: OPT intersect the first job in the chain

Type B1:  $J$  is not type A and  $w(J) > 2w(J_s)$  where  $J_s$  is the last interval in the chain

Type B2:  $J$  is not type A and type B1.

Finally we will sum up over all chains.

# 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  $J_j$ ;  $J_{\text{curr}} \leftarrow J_j$ 
```

---

## Some comments on the algorithm

I haven't read the proof of Fung [2014] for proportional weights but I assume it is implicitly using the predecessor chain concept in Woeginger [1994]. Fung does cite the Woeginger paper. Fung sometimes says his algorithm is in the resumming model but in fact he does also state incorrectly that it is only using revoking.

The idea of revoking  $J_{curr}$  by some  $J_i$  iff  $w(J_i) > 2w(J_{curr})$  occurs in the following papers (and probably other papers):

- 1 The  $1/4$  competitive ratio in Woeginger for interval selection (with C-Benevolent, D-Benevolent and single processing times weighted instance) in the real time model with revoking.
- 2 It also occurs in Garay et al [1997] for the online (with revoking) result obtaining the ratio  $2 + \sqrt{5}$  for call processing (interval selection) problem. This paper does not reference Woeginger.
- 3 It occurs in the Erlebach and Spieksma [2003] priority algorithm (with revoking) for interval election and JISP problems for proportional and arbitrary weight. It is also in Horn's MSc thesis.

## Continuation of the comments on the algorithm

**Comment:** I didn't think a deterministic constant competitive ratio was possible if we only allow revoking and not restarting for proportional weights. But we knew that for online interval selection, while there is no constant ratio for unweighted interval selection in the online model with revoking there is the Garay et al  $2 + \sqrt{5}$  ratio for proportional weights. We have the same juxtaposition of results in the paper we are writing.

**Theorem:** For the unweighted throughput problem, when there are at most  $k$  distinct processing times, there is a deterministic real time algorithm (with revoking) that achieves a  $\frac{1}{2k}$  ratio and there is no such algorithm that achieves a ratio better than  $\frac{1}{k}$ . So there is a gap. We conjecture the correct ratio is  $\frac{1}{2k}$

## High Level Idea of the Competitive Analysis for the C-Benevolent weighted Throughput algorithm.

As already noted, we follow Woeginger's predecessor and charging. A predecessor chain is a sequence of jobs  $J_1, J_2, \dots, J_s$  where  $J_{i+1}$  caused the algorithm to revoke  $J_i$  and  $J_s$  is accepted by the algorithm.

The predecessor chains are disjoint (although we can have a chain  $C$  followed by a chain  $C'$  where  $C'$  starts where  $C$  ends). Therefore we can basically restrict our analysis to a specific chain.

Our goal is to charge the intervals in OPT that intersect a chain to that chain. An OPT interval can only intersect two chains. When it intersects two chains, that may cause us to merge chains.

Let  $p_1, \dots, p_s$  be the processing times for the algorithm in a chain  $C$  and let  $q_1, \dots, q_t$  be the processing times for OPT intervals intersecting the chain. We can conceptually merge OPT intervals and since the weight function is increasing and convex this can only help OPT. We can then assume  $t \leq s$  and add fictional OPT intervals with 0 processing times so that  $t = s$ .

## Finishing the sketch of the analysis

We will want to show that  $\sum_i^k p_i \geq \sum_i^k q_i$  for all  $k \leq s$ .

We can then use Karamata's Inequality to show that  $\sum_i^s w(p_i) \geq \sum_i^s w(q_i)$ .

We will charge OPT intervals  $J$  to Type A: OPT intersect the first job in the chain

Type B1:  $J$  is not type A and  $w(J) > 2w(J_s)$  where  $J_s$  is the last interval in the chain

Type B2:  $J$  is not type A and type B1.

Finally we will sum up over all chains.

## Another result in the paper with Michael and Nadim

We extend the real time model by introducing a new model. Specifically, we assume that every job is given a sufficient amount of advanced notice of an arriving job. If  $r_i$  is the arrival time of job  $J_i$ , the algorithm will know about  $J_i$ 's arrival at time  $a_i$ . A job has  $t$  advance notice if  $r_i - a_i \geq t \cdot p_i$ . That is, the advance notice is at least proportional to the processing time.

We adapted the  $\frac{1}{5}$  algorithm in the text to give an algorithm that *doesn't use revoking* and achieves competitive ratio  $\frac{t-t^2}{1+t-t^2}$  for  $0 < t \leq \frac{1}{2}$ . Hence this modified algorithm achieves the same  $\frac{1}{5}$  ratio as in the text (but now without [preemption]) when  $t \geq \frac{1}{2}$ . It doesn't seem that more advance notice (i.e.  $t > \frac{1}{2}$ ) helps. Of course if  $t = \infty$ , then this becomes an offline problem.

This result is for proportional weights and does not extend to all C-Benvolent weight functions.

Can we use advanced notice for other real-time problems?