Online Throughput Scheduling with Revocation

UNIVERSITY OF TORONTO

Changdao (Michael) He
Supervisor: Allan Borodin

Motivation

- We study **online throughput scheduling** on one machine: jobs arrive over time, and the scheduler must act without knowing future jobs.
- In the **preemption-restart** model, a revoked job can later restart from scratch, and a tight deterministic $1/2$-competitive algorithm is known.
- In the **preemption-revoke** model, a running job may be aborted, but all work done on it is lost forever.
- At first glance, revoke still seems powerful: the scheduler can abandon a bad choice and switch to a new job.
- Our result shows a sharp negative answer: **for deterministic algorithms, revocation is not enough to obtain any constant competitive ratio.**

Problem Model

- Single-machine, unweighted throughput scheduling.

- Objective: maximize the number of jobs completed on time.
- A job is $J_i = (r_i,\ p_i,\ s_i)$, where:
  - r_i: release time
  - $p_i > 0$: processing time
  - $s_i \geq 0$: slack
- A job is feasible only if it starts by $r_i + s_i$ (latest-start constraint). We write deadline $d_i = r_i + p_i + s_i$ only as shorthand for the end of its window $[r_i, d_i)$.

- **Preemption-revoke:** the currently running job may be aborted at any time, but once revoked it is permanently lost and cannot be resumed or restarted.

Performance Measure

- ALG : an online algorithm that only sees jobs when they arrive.

- OPT : an offline optimal schedule that knows the full input sequence in advance.

- We compare ALG to OPT using the ratio
$$\frac{ALG}{OPT}$$
- A **constant competitive ratio** would mean there is a fixed constant $c > 0$ such that
$$\frac{ALG}{OPT} \geq c$$
on every input.

- We show this is impossible: by recursively nesting the adversarial construction, ALG completes at most one job while OPT completes arbitrarily many.

Key Lemma

Side-gap Lemma

Let $J = (r, p, s)$ with $s \geq 2p$ and $d = r + p + s$. If $[x, y) \subset [r, d)$ has length at most p, then at least one of the two side gaps, $[r, x)$ or $[y, d)$, is long enough to schedule J on time.

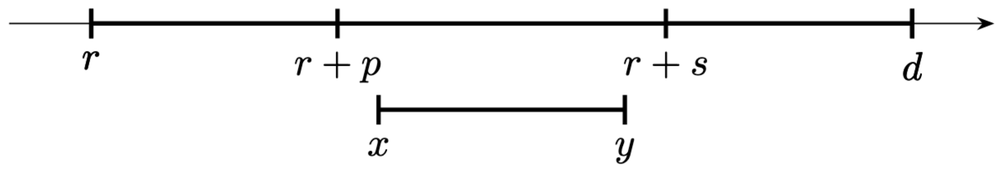

Figure 1: *Intervals $[r, d)$ and $[x, y)$ illustrating the side-gap condition.*

- This is the key tool that preserves OPT feasibility throughout the recursive construction.

Main Result

- The proof starts from a 3-job adversarial construction and then recursively nests jobs to force the competitive ratio arbitrarily close to 0.

Main Theorem

For every $k \geq 3$, there exists an input instance such that ALG completes at most one job while OPT completes at least k jobs. Therefore, no deterministic online algorithm can achieve a constant competitive ratio in the preemption-revoke model.

Inductive Construction

- Assume we already have an instance I_k with jobs $J_1, \dots, J_k$.
- The windows are nested: $[r_{i+1}, d_{i+1}) \subset [r_i, d_i)$, and each new job is released either inside the interval where ALG runs J_i or just after the latest feasible start of J_i.
- Inductive Invariant:
  - ALG completes at most one job in I_k
  - OPT completes all k jobs
  - each job has slack $s_i \geq 2p_i$
- To build I_{k+1}, the adversary adds one more nested job J_{k+1}, placing it according to ALG's behavior on J_k.
  - If ALG starts J_k, release J_{k+1} immediately after that start, fully nested inside the processing interval of J_k.

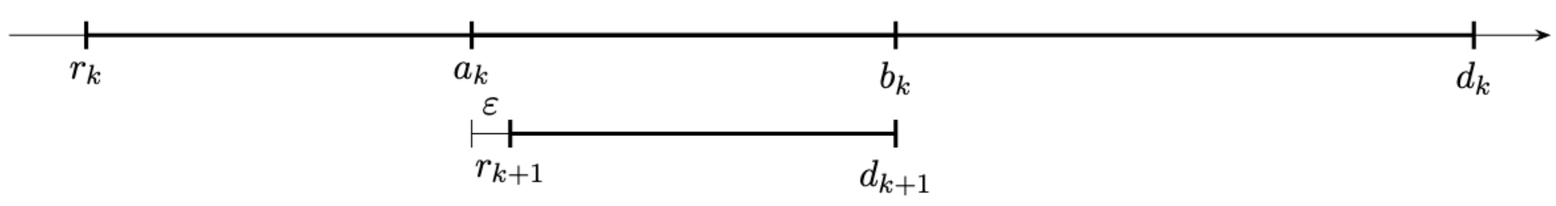

Figure 2: *J_{k+1} is released immediately after a_k*

- If ALG never starts J_k, release J_{k+1} just after the latest feasible start of J_k, making J_k infeasible for ALG.

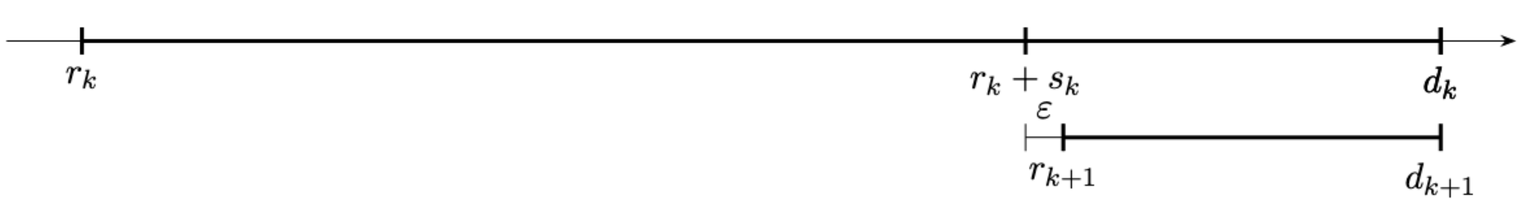

Figure 3: *J_{k+1} is released immediately after $r_k + s_k$ (latest feasible start of J_k)*

- In both branches, the inductive invariant is preserved, so the construction extends from I_k to I_{k+1}; hence $ALG \leq 1$ while $OPT \geq k$, and the competitive ratio can be forced to $1/k \to 0$.

Takeaway and Next Steps

- Even though revocation allows the scheduler to abandon bad choices, it is still not enough to guarantee any deterministic constant competitive ratio.
- A natural next step is to identify restricted settings, such as small slack, or only k distinct processing times, in which the revoke model might still admit a constant competitive ratio.

Preprint: arXiv:2510.15318v1

Contact: changdao.he@mail.utoronto.ca