SODA26: An Optimal Online Algorithm for Robust Flow Time Scheduling, Reading Note

Siyu Liu

January 2026

1 Problem Setting:

- **Flow time scheduling:**
  - Single machine, jobs can be preempted and resumed arbitrarily without loss.
  - Jobs arrive online. Job j arrives at r_j, completes at c_j.
- **Goal:** minimize $\sum_j (c_j - r_j)$
- **Observation:** Suppose at time t, the set of jobs alive is $A(t)$.

$$\sum_j (c_j - r_j) = \sum_t |A(t)|$$

- **Clairvoyant setting:** Job j reports its size p_j upon arrival.
  - Optimal algorithm: SRPT (shortest remaining processing time first)
  - 1-competitive
  - Proof is simple: $|A^{\text{OPT}}(t)| \geq |A^{\text{SRPT}}(t)| \quad \forall t$.
- **Non-clairvoyant setting:** Size is known only upon completion.
  - Randomized: $O(\log n)$ competitive
  - Asymptotically tight.
- **Today's setting: Robust Flow time secheduling**
  - Job j reports a prediction $\hat{p}_j$ of its size upon arrival.

$$\frac{p_j}{\mu_1} \leq \hat{p}_j \leq \mu_2 p_j \quad \forall j$$

  - $\mu := \mu_1 \cdot \mu_2$ is called **distortion**.
  - Zig-Zag algorithm: $O(\mu \log \mu)$ even oblivious to distortion.
  - lower bound: $\Omega(\mu)$ for any randomized algorithm, even known distortion.

2 Main Result:

BALANCE: deterministic algorithm with $O(\mu)$ competitive ratio for robust flow time scheduling.

3 Warm Up: (Primal-Dual analysis for SRPT in clairvoyant setting)

3.1 LP formulation:

Let $x_{j,t}$ = job j is alive at time t. Until time t, the total requirement of processing time of a set of jobs S released before t denoted as $p(S) = \sum_{j \in S} p_j$. Let $r_S := \min_{j \in S} r_j$ and $L_S = t - r_S$.

So at least $e(S,t) = \max\{0, p(S) - L_S\}$ amount of processing remains to be performed on jobs in S.

$$\sum_{j \in S} p_j x_{j,t} \geq \text{remaining processing time of jobs in } S \geq e(S,t)$$

A natural formulation will be:

$$\begin{aligned} \min \quad & \sum_j \sum_{t \geq r_j} x_{j,t} \\ \text{s.t.} \quad & \sum_{j \in S} p_j x_{j,t} \geq e(S,t) \quad \forall S, t \\ & x_{j,t} \geq 0 \end{aligned}$$

Note that it decouples across time.
So to simplify, from now on, we fix a time t^*, and prove for any t^*.
Now the LP becomes

$$\begin{aligned} \min \quad & \sum_{j : r_j \leq t^*} x_{j,t^*} \\ \text{s.t.} \quad & \sum_{j \in S} p_j x_{j,t^*} \geq e(S,t^*) \quad \forall S \\ & x_{j,t^*} \geq 0 \end{aligned}$$

However, this LP has unbounded relaxation gap.
A bad instance will be:

- Only one job, size M.
- At time t^*, $e(\{1\}, t^*) = 1$.
- In the LP, $x_{1,t^*} = \frac{1}{M}$.

- But in the Integer Programming version, $x_{1,t^*} = 1$.
- The gap M is unbounded.

$$\text{Dual}^* = \text{LP}^* \leq \frac{1}{M}\text{IP}^* = \frac{1}{M}\text{OPT}$$

Dual* is unboundedly small, we're impossible to set reasonable dual variables
Fortunately, we can use an idea in knapsack covering relaxation.
We cut off large p_j's since they can at most count as $e(S, t^*)$ in the covering.

$$\begin{aligned}
\text{Primal:} \quad \min \quad & \sum_{j: r_j \leq t^*} x_{j,t^*} \\
\text{s.t.} \quad & \sum_{j \in S} \min\{p_j, e(S, t^*)\} x_{j,t^*} \geq e(S, t^*) \quad \forall S \\
& x_{j,t^*} \geq 0
\end{aligned}$$

$$\begin{aligned}
\text{Dual:} \quad \max \quad & \sum_{S} e(S, t^*) y_{S,t^*} \\
\text{s.t.} \quad & \sum_{S: j \in S} \min\{p_j, e(S, t^*)\} y_{S,t^*} \leq 1 \quad \forall j \text{ with } r_j \leq t^* \\
& y_{S,t^*} \geq 0
\end{aligned}$$

From now on, we hide t^* to simplify notation in $e(S, t^*)$ and y_{S,t^*}.

3.2 Setting Dual Variables

First we introduce a few notations.

- $p_j(t)$ denotes the remaining processing of j at time t.
- $A(t)$ denotes the set of alive jobs at time t.
- $j(t)$ denotes the job chosen to process at time t.
- $q(t)$ denotes the remaining processing of $j(t)$ at time t.

$$\text{i.e. } q(t) = p_{j(t)}(t)$$

By SRPT rule, $q(t) \leq p_j(t)$ for all $j \in A(t)$.

By arguing separately for each maximal busy period, we can assume the machine never idles. So $A(t) \neq \emptyset$, $j(t), q(t)$ are well-defined.

For any $x \geq 0$, let $\tau(x)$ be the earliest time before t^* s.t. the algorithm only processes job with remaining size $\leq x$ in $[\tau(x), t^*]$, i.e. $q(t) \leq x$ for all $t \in [\tau(x), t^*]$.

Define $S(x) = \{\text{jobs of size at most } x \text{ released during } [\tau(x), t^*]\}$.

The intuition here is that we'd like to set non-zero values **only** to those y_S with large $e(S)$ so that we can keep the objective large enough as $\Omega(|A(t^*)|)$ while remaining the solution feasible. We want to find some S that we can lower bound $e(S)$.

Lemma 1.

$$e(S(x), t) \geq \sum_{j: p_j(t) \leq x} p_j(t) - x$$

Proof. Since $\tau(x)$ is the earliest time satisfying the property,

at $\tau(x) - \varepsilon$, all job's remaining size $> x$. (Otherwise, SRPT also only processes jobs with remaining size $\leq x$ in $[\tau(x) - \varepsilon, t^*]$).

So at $\tau(x)$, one of the following two cases must happen:

(1) A new job with size $\leq x$ is released at $\tau(x)$.

(2) Some job's remaining size reduced to x at $\tau(x)$.

For case (1), we know $\min_{j \in S(x)} r_j = \tau(x)$. And $S(x) = \{j : p_j(t) \leq x\}$. So $[\tau(x), t]$ only processes jobs in $S(x)$.

$$e(S(x), t) = \sum_{j: p_j(t) \leq x} p_j(t)$$

For case (2), suppose job j^*'s remaining size reduced to x at $\tau(x)$.

$$\begin{aligned}
e(S(x), t) &= \sum_{j \in S(x)} p_j - \left(t - \min_{j \in S(x)} r_j\right) \\
&\geq \sum_{j \in S_x} p_j - (t - \tau(x)) \\
&= \sum_{j \in S_x} p_j + p_{j^*}(t) - p_{j^*}(t) - (\text{time spent on } S_x + \text{time spent on } j^*) \\
&= \sum_{j \in S_x} (p_j - \text{time spent on } j) + p_{j^*}(t) - (p_{j^*}(t) + \text{time spent on } j^*) \\
&= \sum_{j \in S_x \cup \{j^*\}} p_j(t) - x \\
&= \sum_{j: p_j(t) \leq x} p_j(t) - x
\end{aligned}$$

□

To better use this inequality, consider the following case:

Sort all alive jobs according to their remaining size at time t^*. Denote them as $p_1(t^*) \leq p_2(t^*) \leq \cdots \leq p_{|A(t)|}(t^*)$.

Then

$$e(S_{p_i(t^*)}) \geq \sum_{j:p_j(t^*)\leq p_i(t^*)} p_j(t^*) - p_i(t^*) = \sum_{j=1}^{i-1} p_j(t^*)$$

We'd like to set non-zero values only to those $S_{p_i(t^*)}$'s.

We still need a **bucketing technique**.
Say a job j is in bucket B_k if $p_j(t^*) \in [2^k, 2^{k+1})$.
Suppose j_k is the job with the largest remaining size in B_k.
Denote $S_k = S(p_{j_k}(t^*))$ to simplify notation.
Then immediately:

$$\begin{aligned} e(S_k) &\geq \sum_{j:p_j(t^*)\leq p_{j_k}(t^*)} p_j(t^*) - p_{j_k}(t^*) \\ &\geq \sum_{j\in B_k, j\neq j_k} p_j(t^*) \\ &\geq (|B_k| - 1)\cdot 2^k \end{aligned}$$

Now we set dual variables.

- **Case 1:** If $|B_k| = 1$ and k is not the first non-empty bucket.

$$\text{set } y_{S_k} = \frac{1}{e(S_k)}$$

- **Case 2:** If $|B_k| \geq 2$, set $y_{S_k} = \frac{1}{2^k}$.
- The other $y_S = 0$.

Lemma 2.

$$\sum_S y_S e(S) \geq \frac{|A(t^*)| - 1}{2}$$

Proof. For S_k in Case 1:

$$\sum_{S_k \text{ in case 1}} y_S e(S) = |\{k : |B_k| = 1\}| - 1$$

For S_k in Case 2:

$$\begin{aligned} \sum_{S_k \text{ in case 2}} y_S e(S) &\geq \sum_{S_k \text{ in case 2}} \frac{1}{2^k}\cdot(|B_k| - 1)\cdot 2^k \\ &\geq \sum_{B_k:|B_k|\geq 2} \frac{|B_k|}{2} \end{aligned}$$

Summing up:

$$\sum_S y_S e(S) \geq \frac{\sum_k |B_k| - 1}{2} = \frac{|A(t^*)| - 1}{2}$$

□

Lemma 3. *$\{y_S/5\}$ is feasible dual variables.*

Proof. $\forall$ job j, suppose $p_j \in [2^k, 2^{k+1})$.

Only $S_{k'}$ with $k' \geq k$ can possibly contain j.

For $S_{k'}$ in Case 2, which is easier:

$$\sum_{S_{k'} \text{ in case 2}} \min\{p_j, e(S_{k'})\} \cdot y_{S_{k'}} \leq \sum_{S_{k'} \text{ in case 2}} p_j \cdot \frac{1}{2^{k'}}$$
$$\leq \sum_{S_{k'} \text{ in case 2}} \frac{2^{k+1}}{2^{k'}}$$

For $S_{k'}$ in Case 1,

It's not enough to bound just by $\min\{p_j, e(S_{k'})\}y_{S_{k'}} \leq e(S_{k'})y_{S_{k'}} \leq 1$.

Notice if k' is not the smallest index in Case 1, and $k'' < k'$ is also in Case 1, then

$$e(S_{k'}) \geq \sum_{j: p_j(t^*) < p_{j_{k'}}(t^*)} p_j(t^*)$$
$$\geq \sum_{j \in B_{k''}} p_j(t^*) \geq 2^{k''}$$

Since in Case 1, $|B_{k''}| = 1$, and the job size is at least $2^{k''}$.

Thus,

$$\sum_{S_{k'} \text{ in Case 1}} \min\{p_j, e(S_{k'})\}y_{S_{k'}} \leq 1 + \sum_{k' \text{ in Case 1}} \frac{2^{k+1}}{2^{k''}} \quad \text{(where } k'' < k' \text{ is the previous Case 1 index)}$$
$$\leq 1 + \sum_{k'' \text{ in Case 1}} \frac{2^{k+1}}{2^{k''}}$$

The "1" comes from the smallest index in Case 1.

Summing up:

$$\sum_{S_{k'}} \min\{p_j, e(S_{k'})\}y_{S_{k'}} \leq 1 + \sum_{k' > k} \frac{2^{k+1}}{2^{k'}} \leq 5.$$

□

Theorem 1. *SRPT is 11-competitive.*

Proof. For any t^*, combining Lemma 2 and Lemma 3:

$$\frac{|A(t^*)| - 1}{2} \leq 5 \cdot \text{Dual} \leq 5 \cdot \text{Dual}^* = 5 \cdot \text{Primal}^* \leq 5 \cdot \text{OPT}(t^*)$$

Summing over t^*:

$$\frac{\text{ALG} - \sum p_j}{2} \leq 5 \cdot \text{OPT}$$

$$\text{ALG} \leq 10 \cdot \text{OPT} + \sum p_j$$

Since $\text{OPT} \geq \sum p_j$:

$$\text{ALG} \leq 11 \cdot \text{OPT}$$

□

4 Robust Flow-time scheduling:

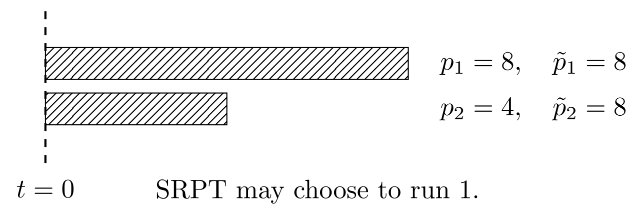
4.1 What about SRPT?

Always run the job with shortest **estimated** remaining size.

We can give arbitrarily bad instance even for $\mu = 2$.

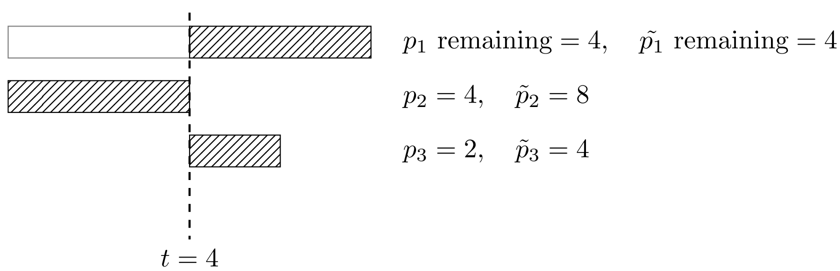
---

1. At time $t = 0$:



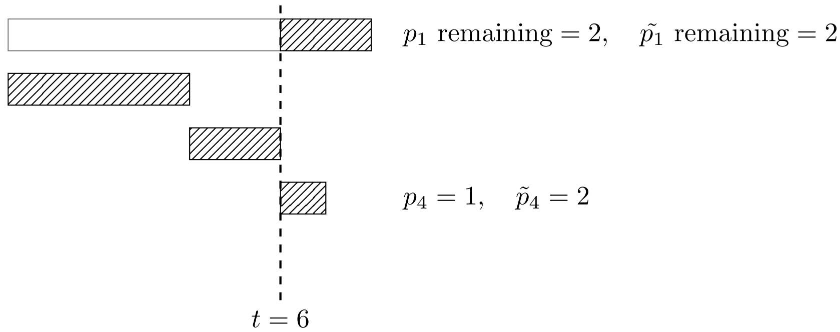
SRPT may choose to run 1.

2. At time $t = 4$:



SRPT may choose to run 1.

3. At time $t = 6$:

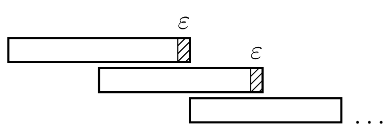


SRPT may choose to run 1.

At time 7, no jobs are accomplished. **4 jobs alive**.
However, OPT can make just **1 job alive** (by finishing Job 2, 3, 4 first).
This can be generalized to be arbitrarily bad.

4.2 SJF

SJF (Shortest Job First) can handle this bad instance.
Always process the job with minimum $\tilde{p}_j$.
However, arbitrarily bad instance exists even for exact estimate ($\mu = 1$).



SJF would finish no job.
OPT can make only one job alive.

We want to find a balance:

- Prefer jobs with small estimated processing time. (Conquer hard instance for SRPT)
- Avoid creating too many partially processed jobs. (Conquer hard instance for SJF)

4.3 Algorithm (BALANCE):

Denote:

- $A^{\text{full}}(t)$: the jobs not processed at all at time t.
- $A^{\text{partial}}(t)$: the jobs partially processed at time t.
- $A(t)$: all jobs alive at time t.

Classify all jobs into classes:

$$j \text{ is in class } k \text{ if } \tilde{p}_j \in [2^k, 2^{k+1})$$

Maintain a priority queue **Queue** and a stack **Stack**.

The priority in **Queue** is by decreasing job class, i.e., the smallest job classes are at front. Jobs in the same class are arbitrarily ordered.

Queue is used to put **full** jobs.

Stack is used to put **partial** jobs.

At each time t,

- If some job arrives, put it into **Queue** at the correct position.
- Next, if one of the following holds:
  (a) **Stack** is empty, or
  (b) The job front(Queue) has strictly smaller class than the job top(Stack) (*in fact, equivalently strictly smaller than all partial jobs, as we'll prove later*) and $|A^{\text{full}}(t)| \geq \frac{|A(t)|}{4}$.

  , then move front(Queue) from **Queue** to top of **Stack**.
- Now, process top(Stack).

4.4 Analysis (Primal Dual Scheme)

We begin with proving several basic properties of the algorithm.

Lemma 4. *At any time t, **Stack** is strictly monotone in job class, i.e. the job at top is of strictly smaller class than job below.*

Proof. Since each time the stack adds elements, it adds a job of strictly smaller class than top(Stack) at top. □

Lemma 5. *If a job j moves from **Queue** to **Stack** at time t, it belongs to the smallest nonempty class in $A(t)$.*

Proof. Since j must be front(Queue) and of smaller class than all jobs in **Stack**. □

Lemma 6. *Suppose a job j of class k is processed at time t. If there's a job of strictly smaller class than k in $A(t)$, then there cannot be another job of class at most k in $A(t)$.*

Proof. Let j_1 be a job of class $< k$ in $A(t)$.

Note that j_1 must be in **Queue**.

Then j must be already at top of **Stack**, not the one moved from **Queue** to **Stack** at t.

The reason why j_1 is not moved from **Queue** to **Stack** must be

$$|A^{\text{full}}(t)| < \frac{|A(t)|}{4}$$

So j cannot be the only job in **Stack**. Suppose j' is the job immediately below j in **Stack**.

Towards a contradiction, assume there's a job $j_2 \neq j_1$ of class $\leq k$ in $A(t)$. $j_2 \in A^{\text{full}}(t)$ similarly.

Denote t' as the first time job j is processed, i.e. moved from **Queue** to **Stack**.

Let J' be the jobs in $A(t)$ that are released after t'.

Claim: All jobs in J' except j must be full at time t.

This is because if some job in J' becomes partial, it would be placed above j' in **Stack**, which contradicts with j' is immediately below j.

At time t', since j' is moved from **Queue** to **Stack**,

$$|A^{\text{full}}(t')| \geq \frac{1}{4}|A(t')|$$

In $[t', t]$, $|A^{\text{full}}(t')|$ adds by $|J'|$ arrival and minuses by 2 due to the moving of j and j'.

Since $j, j_1, j_2 \in J'$, we have $|J'| \geq 3$.

$$|J'| - 2 \geq \frac{1}{4}|J'|$$

So $|A^{\text{full}}(t)| \geq \frac{1}{4}|A(t)|$. Contradiction. □

Now we recall the LP. We continue to prove for any fixed t^*.

$$\begin{aligned}
\text{Primal:} \quad & \mathit{min} \quad && \sum_{j: r_j \leq t^*} x_j \\
& \text{s.t.} && \sum_{j \in S} \min\{p_j, e(S)\} x_j \geq e(S) \quad \forall S \\
& && x_j \geq 0
\end{aligned}$$

$$\begin{aligned}
\text{Dual:} \quad & \max \quad && \sum_{S} e(S) y_S \\
& \text{s.t.} && \sum_{S: j \in S} \min\{p_j, e(S)\} y_S \leq 1 \quad \forall j \text{ with } r_j \leq t^* \\
& && y_S \geq 0
\end{aligned}$$

Recall we'd like to give lower bound of $e(S)$ for specific S.

Denote $A_k^{\text{full}}(t^*)$ be the full jobs of class (k) at t^*.

Lemma 7. *Suppose $A_k^{full}(t^*)$ is non-empty.*

Let t_0 be the last timestep before t^* when a job j of class $> k$ is processed.

Let S be set of jobs of class $\le k$ released during $[t_0, t^*]$.

Then,

$$e(S) \ge \sum_{k' \le k} \sum_{j \in A_{k'}^{full}(t^*)} p_j - \mu_2 \cdot 2^{k+1}$$

Proof. During $[t_0, t]$, we only process jobs of class $\le k$.

By Lemma 6, at most one job of class $\le k$ in $A(t_0)$ (since either $A(t_0) = \emptyset$ or there's a job $\ge k+1$ being processed at t_0).

This job has estimated size $\le 2^{k+1}$. So actual size $\le \mu_2 \cdot 2^{k+1}$.

$$\begin{aligned} e(S) &\ge \sum_{j \in S} p_j(t^*) - \mu_2 \cdot 2^{k+1} \\ &\ge \sum_{j \in S,\ j \text{ is full}} p_j - \mu_2 \cdot 2^{k+1} \\ &= \sum_{k' \le k} \sum_{j \in A_{k'}^{\text{full}}(t^*)} p_j - \mu_2 \cdot 2^{k+1} \end{aligned}$$

□

Unfortunately this works well when $|A_k^{\text{full}}(t^*)|$ is large, not enough when $|A_k^{\text{full}}(t^*)|$ is small.

We introduce:

Property 1 (Reduction Property (RP))**.** *Every job $j \in A^{full}(t^*)$ satisfies:*

$$p_j \le \max_{t \in [r_j, t^*)} 2^{q(t)+1}$$

where $q(t)$ is the job class being processed at t.

We'll reduce any instance I to have the property later, but now let's assume this.

Lemma 8. *Suppose $A_k^{full}(t^*)$ is non-empty. Let t_0 be the last time when a job of class $\ge k$ is processed. Let S be the jobs of class $< k$ released during $[t_0 + 1, t^*]$. Then*

$$e(S) \ge \sum_{k' < k} \sum_{j \in A_{k'}^{full}(t^*)} p_j$$

Proof. During $[t_0+1, t^*]$, we only process jobs of class $< k$.

By RP (Reduction Property), any job in $A^{\text{full}}(t^*)$ that is released during $[t_0+1, t^*]$ must have job class $< k$.

Take $j \in A_k^{\text{full}}(t^*)$, then $r_j \le t_0$. So $j \in A_k^{\text{full}}(t_0)$.

By Lemma 6, there's no job of class $< k$ in $A(t_0)$.

So any job of class $< k$ processed during $[t_0+1, t^*]$ must be released also during $[t_0+1, t^*]$.

$$\begin{aligned} e(S) &= \sum_{j\in S} p_j(t^*) \\ &\ge \sum_{j\in S, j \text{ full}} p_j \\ &= \sum_{k'<k} \sum_{j\in A_{k'}^{\text{full}}(t^*)} p_j \end{aligned}$$

□

4.5 Setting Dual Variables

For each k s.t. $A_k^{\text{full}}(t^*)$ is non-empty. Let $S_0(k), S_1(k)$ be the sets given by Lemma 7 and Lemma 8 respectively.

- **Case 1.** If $|A_k^{\text{full}}(t^*)| < 3\mu$, set $y_{S_0(k)} := \frac{|A_k^{\text{full}}(t^*)|}{3\mu\cdot e(S_0(k))}$
- **Case 2.** If $|A_k^{\text{full}}(t^*)| \ge 3\mu$, set $y_{S_0(k)} = \frac{1}{\mu_2\cdot 2^k}$.
- All the other y_S are set to 0.

Let I_0, I_1 denote the indices k satisfying Case 1 and Case 2 respectively.

Lemma 9.

$$\sum_S e(S)y_S \ge \frac{|A^{full}(t^*)|}{3\mu} \ge \frac{\frac{1}{4}|A(t^*)|-1}{3\mu}$$

Proof. For $k \in I_0$:

$$\sum_{k\in I_0} e(S_0(k))y_{S_0(k)} = \sum_{k\in I_0} \frac{|A_k^{\text{full}}(t^*)|}{3\mu}$$

For $k \in I_1$:

$$\begin{aligned}
e(S_0(k)) &\geq \sum_{k' \leq k} \sum_{j \in A_{k'}^{\text{full}}(t^*)} p_j - \mu_2 \cdot 2^{k+1} \\
&\geq \sum_{j \in A_k^{\text{full}}(t^*)} p_j - \mu_2 \cdot 2^{k+1} \\
&\geq |A_k^{\text{full}}(t^*)| \cdot \frac{2^k}{\mu_1} - \mu_2 \cdot 2^{k+1} \\
&= (|A_k^{\text{full}}(t^*)| - 2\mu) \cdot \frac{2^k}{\mu_1} \\
&\geq \frac{|A_k^{\text{full}}(t^*)|}{3\mu_1} \cdot 2^k \quad (\text{since } |A_k| \geq 3\mu)
\end{aligned}$$

So,

$$\sum_{k \in I_1} e(S_0(k)) y_{S_0(k)} \geq \sum_{k \in I_1} \frac{|A_k^{\text{full}}(t^*)|}{3\mu_1} \cdot 2^k \cdot \frac{1}{\mu_2 \cdot 2^k} \geq \sum_{k \in I_1} \frac{|A_k^{\text{full}}(t^*)|}{3\mu}$$

Summing up,

$$\sum_S e(S) y_S \geq \frac{|A^{\text{full}}(t^*)|}{3\mu}$$

It remains to prove $\forall t, |A^{\text{full}}(t)| \geq \frac{|A(t)|}{4} - 1$.
This is because each time a full job becomes partial:

$$|A^{\text{full}}(t)| \geq \frac{|A(t)|}{4}$$

After moving the job from full to partial,

$$|A^{\text{full}}(t)| \geq \frac{|A(t)|}{4} - 1$$

The other time $|A^{\text{full}}(t)|, |A(t)|$ both increases by 1 (upon job arrival), which maintains the inequality. $\square$

Now we establish approximate dual feasibility.

For any job j^*, suppose it's of class k^*. Any $S_0(k)$ or $S_1(k)$ containing j^* must satisfy $k \geq k^*$.

Define:

$$\begin{aligned}
I_0(j^*) &:= \{k \geq k^* : k \in I_0, j^* \in S_0(k)\} \\
I_1(j^*) &:= \{k \geq k^* : k \in I_1, j^* \in S_1(k)\}
\end{aligned}$$

Lemma 10.

$$\sum_{k \in I_0(j^*)} \min\{e(S_0(k)), p_{j^*}\} y_{S_0(k)} < 10$$

Proof. Define $T_\ell := \{k \in I_0(j^*) : |A_k^{\text{full}}(t^*)| \in [2^\ell, 2^{\ell+1})\}$.

Since $\forall k \in I_0(j^*)$, $|A_k^{\text{full}}(t^*)| < 3\mu$, we have:

$$\ell \leq \Gamma := \lfloor \log_2(3\mu) \rfloor$$

$$\begin{aligned}
\sum_{k \in I_0(j^*)} \min\{e(S_0(k)), p_{j^*}\} y_{S_0(k)} &= \sum_{k \in I_0(j^*)} \min\{e(S_0(k)), p_{j^*}\} \frac{|A_k^{\text{full}}(t^*)|}{3\mu e(S_0(k))} \\
&\leq \sum_{\ell=0}^{\Gamma} \sum_{k \in T_\ell} \min\{e(S_0(k)), p_{j^*}\} \cdot \frac{2^{\ell+1}}{3\mu e(S_0(k))} \\
&\leq \sum_{\ell=0}^{\Gamma} \frac{2^{\ell+1}}{3\mu} \sum_{k \in T_\ell} \min\left\{1, \frac{p_{j^*}}{e(S_0(k))}\right\} \\
&\leq \sum_{\ell=0}^{\Gamma} \frac{2^{\ell+1}}{3\mu} \underbrace{\sum_{k \in T_\ell} \min\left\{1, \frac{2^{k^*+1} \cdot \mu_2}{\sum_{k'<k} \sum_{j \in A_{k'}^{\text{full}}(t^*)} p_j}\right\}}_{(*)}
\end{aligned}$$

Rearrange indices in T_ℓ in increasing order as $k_1, k_2, \ldots, k_{|T_\ell|}$.

$$\begin{aligned}
(*) &= \sum_{i=1}^{|T_\ell|} \min\left\{1, \frac{\mu_2 \cdot 2^{k^*+1}}{\sum_{k'<k_i} \sum_{j \in A_{k'}^{\text{full}}(t^*)} p_j}\right\} \\
&\leq 1 + \sum_{i=2}^{|T_\ell|} \frac{\mu_2 \cdot 2^{k^*+1}}{\sum_{j \in A_{k_{i-1}}^{\text{full}}(t^*)} p_j} \\
&\leq 1 + \sum_{i=2}^{|T_\ell|} \frac{\mu_2 \cdot 2^{k^*+1}}{2^\ell \cdot \frac{2^{k_{i-1}}}{\mu_1}} \\
&\leq 1 + \mu \cdot \frac{2^{k^*+2}}{2^\ell \cdot 2^{k_1}}
\end{aligned}$$

Let $K_\ell \geq k^*$ be the smallest index in T_ℓ.

$$\begin{aligned}
\sum_{\ell=0}^{\Gamma} \frac{2^{\ell+1}}{3\mu} \sum_{k \in T_\ell} \min\left\{1, \frac{2^{k^*+1} \cdot \mu_2}{\sum_{k'<k} \sum_{j \in A_{k'}^{\text{full}}(t^*)} p_j}\right\} &\leq \sum_{\ell=0}^{\Gamma} \mathbb{1}[T_\ell \neq \emptyset] \cdot \frac{2^{\ell+1}}{3\mu} \left(1 + \mu \cdot \frac{2^{k^*+2}}{2^\ell \cdot 2^{K_\ell}}\right) \\
&= \sum_{\ell=0}^{\Gamma} \frac{2^{\ell+1}}{3\mu} + \frac{2^{k^*+3}}{3} \sum_{\ell=0}^{\Gamma} \left(\mathbb{1}[T_\ell \neq \emptyset] \cdot \frac{1}{2^{K_\ell}}\right) \\
&\leq \frac{2^{\Gamma+2}}{3\mu} + \frac{2^{k^*+3}}{3} \cdot \frac{2}{2^{k^*}} \\
&< 10
\end{aligned}$$

Since all $K_\ell \geq k^*$ and are different. □

Lemma 11.
$$\sum_{k\in I_1(j^*)} \min\{e(S_1(k)), p_{j^*}\} y_{S_1(k)} < 4$$

Proof.
$$\begin{aligned}\sum_{k\in I_1(j^*)} \min\{e(S_1(k)), p_{j^*}\} y_{S_1(k)} &\le \sum_{k\ge k^*} p_{j^*}\cdot \frac{1}{\mu_2\cdot 2^k}\\ &\le \sum_{k\ge k^*} \mu_2\cdot 2^{k^*+1}\cdot\frac{1}{\mu_2\cdot 2^k}\\ &< 4\end{aligned}$$
□

4.6 Reduction Property

It remains to reduce any instance $\mathcal{I}$ to satisfy RP. Let $\mathcal{I}^{\text{red}}$ denote the reduced instance to be constructed.

Its job set is exactly the same as the jobs set released before t^* in $\mathcal{I}$, with release dates unchanged.

However, the job size may be different.

$$p_j^{\text{red}} := \min\left\{p_j, \max_{t\in[r_j,t^*)} 2^{q(t)+1}\right\}$$

where $q(t)$ is the job class being processed in $\mathcal{I}$.

Note that, $p_j^{\text{red}} \le p_j$ for all j. Furthermore,

Lemma 12. *if $j \notin A^{full}(t^*)$, $p_j^{red} = p_j$.*

Proof. Since if $j \notin A^{\text{full}}(t^*)$, then j is at least processed once during $[r_j, t^*)$.

$$\max_{t\in[r_j,t^*)} 2^{q(t)+1} \ge 2^{\text{class}(j)+1} > p_j$$

$$p_j^{\text{red}} = p_j$$

□

Lemma 13.
$$OPT_{t^*}(\mathcal{I}^{red}) \le OPT_{t^*}(\mathcal{I})$$

Proof. Since $p_j^{\text{red}} \le p_j$, the OPT scheduling for $\mathcal{I}$ can be copied to use for $\mathcal{I}^{\text{red}}$.

$$OPT_{t^*}(\mathcal{I}^{\text{red}}) \le OPT_{t^*}(\mathcal{I})$$
□

Remark 1. *We point out that BALANCE algorithm is non-deterministic. Since the order of jobs with same class in* ***Queue*** *is arbitrary, the behavior of the algorithm is non-deterministic. All the conclusions we proved above hold actually for all non-deterministic choices. However the next lemma states the existence of non-deterministic choice with certain property.*

Lemma 14. *There exists a choice of non-deterministic behaviour on $\mathcal{I}^{red}$ s.t. $\forall t \leq t^*$, at time t:*

- *The job processed when run on $\mathcal{I}$ is the same as the job processed when run on $\mathcal{I}^{red}$.*
- $A(t) = A^{red}(t)$.
- $A^{full}(t) = A^{full,\ red}(t)$.
- *Stack* $=$ *Stack*^{red} *at time* t.

Proof. Induction on t. $t = 0$, hypothesis holds.

Suppose the hypothesis holds until time t.

- **If no job was released and no job moves from Queue to Stack:** Then hypothesis still holds for $t+1$. Since the jobs being processed in two instances are partial, thus have the same size. They are simultaneously alive or finished.
- **If some job j was released, but no job moves from Queue to Stack:**

$$A^{\text{full}}(t+1) = A^{\text{full}}(t) \cup \{j\} = A^{\text{full, red}}(t) \cup \{j\} = A^{\text{full, red}}(t+1)$$

  The others also keep equal for similar reasons.

- **If some job j moves from Queue to Stack:** Then $j \notin A^{\text{full}}(t^*)$ (since it transitions to partial state). Thus $p_j^{\text{red}} = p_j$.

  We claim j also will be moved from $\text{Queue}^{\text{red}}$ to $\text{Stack}^{\text{red}}$.

  First, j also has minimum job class in $A^{\text{full, red}}(t)$. Indeed, for $j' \in A^{\text{full, red}}(t)$, then $t \in [r_{j'}, t^*)$. Since $q(t) = \text{class}(j)$, we have $p_{j'}^{\text{red}} \geq p_j$.

  Next, since Stack, $A(t)$, $A^{\text{full}}(t)$ are all the same between two instances, the condition that decides whether to move holds both true or false.

  So if we choose j to be $\text{front}(\text{Queue}^{\text{red}})$, which is a legal non-deterministic choice, j will also be processed at $t+1$. Stack, $A(t+1)$, $A^{\text{full}}(t+1)$ also keep the same.

$\square$

4.7 Putting together

Theorem 2. *BALANCE is $O(\mu)$-competitive.*

Proof. Consider instance $\mathcal{I}$.

For all fixed t^*, Construct corresponding reduced instance $\mathcal{I}^{\text{red}}$ and its non-deterministic choice keeping $|A^{\text{red}}(t^*)| = |A(t^*)|$ by Lemma 14.

By Lemma 9, Lemma 10, Lemma 11,

$$|A^{\mathrm{red}}(t^*)| \leq 4 + O(\mu)\mathrm{opt}_{t^*}(\mathcal{I}^{\mathrm{red}})$$

Then, by Lemma 13, Lemma 14,

$$|A(t^*)| \leq 4 + O(\mu)\mathrm{opt}_{t^*}(\mathcal{I})$$

Summing over t^*:

$$\begin{aligned} \mathrm{ALG} &\leq 4\sum_j p_j + O(\mu) \cdot \mathrm{OPT} \\ &\leq O(\mu) \cdot \mathrm{OPT} \end{aligned} \qquad \square$$