Capacity Provisioning Motivated Online Non-Convex Optimization Problem with Memory and Switching Cost

Rahul Vaze, Jayakrishnan Nair

***Abstract*—An online non-convex optimization problem is considered where the goal is to minimize the flow time (total delay) of a set of jobs by modulating the number of active servers, but with a switching cost associated with changing the number of active servers over time. Each job can be processed by at most one fixed speed server at any time. Compared to the usual online convex optimization (OCO) problem with switching cost, the objective function considered is non-convex and more importantly, at each time, it depends on all past decisions and not just the present one. Competitive algorithms are derived for the worst-case inputs.**

I. Introduction

We consider a canonical capacity provisioning problem for data centers where jobs arrive over time and the QoS metric of interest is the flow-time (sum of the delay seen by each job). Each server has a fixed (unit) speed and can serve at most one job at a time. Changing the number of active servers across time slots incurs cost that is modelled as a switching cost and the goal is to minimize the flow time plus the overall switching cost, where the decision variable at each time is the number of active servers.

With flow time $F = \sum_t n(t)$, where $n(t)$ is the number of outstanding jobs at time t, and $s(t)$ is the number of active servers at time t, formally the problem is

$$\min_{s(t)} \sum_t n(t) + \alpha \sum_t c(s(t), s(t-1)) + \theta \sum_t s(t), \quad (1)$$

where $n(t) = \max\left\{\sum_{\tau \le t} \text{arr}_\tau - \sum_{\tau \le t-1} \text{dep}_\tau, 0\right\}$, and arr_τ and dep_τ are the number of jobs that arrive at the start of slot τ and depart at the end of slot τ, respectively, and $\alpha, \theta > 0$ are the weighting parameters for switching and energy cost, respectively.

Problem 1 is similar to the online convex optimization with switching cost (OCO-S) problem introduced in [18], where at each discrete time t, a convex cost function $f_t : \mathbb{R}^{\mathsf{d}} \to \mathbb{R}^+$ is revealed, and an algorithm has to choose an action x_t knowing all $f_\tau, \tau \le t$. The cost of choosing action x_t at time t is the sum of the cost $f_t(x_t)$, and the switching cost $c(x_{t-1}, x_t)$. The goal is to choose $x_t, 1 \le t \le T$ that solves

$$\min_{x(t)} \sum_{t=1}^{T} f_t(x_t) + c(x_{t-1}, x_t). \quad (2)$$

Problem (2) was motivated by capacity provisioning problem [18] where demand $d(t)$ at time t is revealed causally over time, and to serve demand $d(t)$, $s(t)$ number of servers are employed. Changing $s(t)$ over time to dynamically serve $d(t)$ keeps the QoS cost small but incurs a large switching cost. With this motivation, Problem (2) was formulated.

Problem 1 and Problem 2 have two fundamental differences: $f_t(.)$ in (2) only depends on the current action x_t and is convex, while n_t in (1) depends on all the actions $s(\tau), \tau \le t-1$ and new arrivals at time t, and Problem 1 is non-convex since the number of servers and the number of outstanding jobs at any time are both integers.

From a QoS (capacity provisioning) point of view, Conceptually, the principal difference between Problem 1 and Problem 2 is the time scale under consideration. Problem 2 takes an averaged-out view that a time slot is long enough and all the queuing dynamics are assumed to play out within it, while Problem 1 takes a zoomed-in view and are interested in modelling the queuing dynamics explicitly and study its interaction with the switching cost. From an applications point of view, our approach is valid for short time scale settings where dynamics evolve at a rapid rate. Analytically, the difficulty is because of the introduction of memory in the cost functions across time as well as the non-convexity of the cost function.

In this paper, we consider the worst-case input and the objective is to design online (that have access to only causal information) algorithms for solving Problem (1) with small competitive ratios that is defined as follows.

The competitive ratio for an online algorithm $\mathcal{A}$ is

$$\text{cr}_{\mathcal{A}} = \max_{\sigma} \frac{C_{\mathcal{A}}(\sigma)}{C_{\mathsf{OPT}}(\sigma)},$$

where σ is the input, $C_{\mathcal{A}}$ (1) is the cost of an online algorithm $\mathcal{A}$, while C_{OPT} denotes the cost of the optimal offline algorithm OPT, that knows the input σ ahead of time, non-causally. The stochastic model is also studied but for lack of space is moved to the full version [25].

A. Prior Work

1) OCO-S: For the OCO-S (2) with quadratic switching cost, i.e., $c(x_t, x_{t-1}) = ||x_t - x_{t-1}||^2$, if the functions f_t's are just convex, then the competitive ratio of any deterministic online algorithm is unbounded [12]. Thus, starting with [9], to solve (2), f_t's were assumed to be strongly convex.

With μ-strongly convex f_t's, online balanced descent (OBD) algorithm that balances the suboptimality and switching cost at each time, has a competitive ratio of at most $3 + \mathcal{O}(\frac{1}{\mu})$ [13]. Surprisingly, [26] showed that the same guarantee can

Rahul Vaze acknowledges the support of the Department of Atomic Energy, Government of India, under project no. RTI4001.

ISBN 978-3-903176-73-7 © 2025 IFIP

be obtained by a simple algorithm that chooses x_t as the local optimizer of f_t disregarding the switching cost. This result was improved in [12] using a modification of OBD to get a competitive ratio $\mathcal{O}(\frac{1}{\sqrt{\mu}})$ as $\mu \to 0^+$, which is also shown to be the best possible (order-wise) competitive ratio for all possible deterministic online algorithms [12].

Similar results have been obtained when f_t's are locally polyhedral in [9]. Very recently, [6] studied OCO-S under some assumptions on the sequence of minimizers of f_t's and provided an algorithm that works for both the worst case as well as the stochastic input. The OCO-S has also been studied in the information setting of OCO recently [22], where action x_t is chosen before f_t is revealed.

OCO-S with linear switching cost, where $c(x_t, x_{t-1}) = |x_t - x_{t-1}|$ has also been widely considered. For this case, an optimal offline algorithm was derived in [18], while online algorithms have been explored in [18], [17], [3], [5], [1], [9], [4], [10] when f_t's are just convex. For $\mathsf{d} = 1$, the best known online algorithm has the competitive ratio of 2 which matches the lower bound [1]. For general d, when f_t's are convex, the competitive ratio of any algorithm has a lower bound $\Omega(\sqrt{\mathsf{d}})$ [9]. Imposing that f_t's are μ-strongly convex and L-smooth, defining $\kappa := L/\mu$ as the condition number, [4] proposed a constrained online balanced descent (COBD) algorithm with a competitive ratio $\mathcal{O}(\sqrt{\kappa})$. Recently, [15] considered the OCO-S with linear switching cost under some long-term constraints. OCO-S with both trusted and untrusted predictions has also been considered in [7], [8], [16], [11], [21]. A review on OCO-S can be found in textbook [24].

OCO-S with memory has also been studied [23] where memory is introduced only in the switching cost, i.e. $f_t(x^t) = f_t(x_t)$ but switching cost $c(s_t, s_{t-1}) = ||s_t - \sum_{k=1}^{p} \gamma_k s_{t-k}||^2$, i.e. it depends on the last p actions when f_t are strongly convex.

2) OCO with memory: When there is no switching cost, Problem 2 (known as OCO) is studied under the model that action x_t is chosen before function f_t is revealed. OCO with memory has also been studied, i.e., f_t's do not just depend on x_t but also on past decisions. In particular, [2] considered p-length memory cost functions where $f_t(x^t) = f_t(x_t, x_{t-1}, \ldots, x_{t-p+1})$, i.e. the current cost function depends on the last p actions and showed that the FTRL algorithm has a static regret[1] of $O(p^{3/2}\sqrt{T})$, which has been improved in [14] to $O(p\sqrt{T})$, which is also the best possible.

In context of OCO with memory, Problem 1 has unbounded memory $p = \infty$. We overcome the lower bound of $\Omega(p\sqrt{T})$ from [14] on the static regret of any online algorithm for OCO with length p memory by carefully exploiting the structure of Problem 1.

B. Our Contributions

With the worst case input, we assume that all job sizes are identical. Notably, Problem 1 still remains highly challenging. Our contributions are as follows. At time t, let $n(t)$ and $s(t)$ denote the number of outstanding jobs and the number of active servers, respectively.

- When the switching cost is linear and $\alpha \leq 1$, we show that a simple algorithm that chooses the number of active servers equal to the number of outstanding jobs ($s(t) = n(t)$) achieves a competitive ratio of 2. We also construct an input instance where this competitive ratio guarantee is tight for this algorithm.
- When the switching cost is linear and $\alpha > 1$, we show that natural extensions of algorithms [18], [17], [3], [5], [1], [9], [4] known to solve OCO-S with linear switching cost have unbounded competitive ratios. Next, we show that an algorithm that chooses $s(t) = \frac{n(t)}{\alpha^{1/4}}$ has a competitive ratio of $O(\alpha^{1/4})$ and a matching lower bound for any algorithm in the class $s(t) = \frac{n(t)}{\alpha^{\gamma}}$ for any $\gamma > 0$. This establishes that the considered problem in the linear swiching cost case has a very different characterization than OCO-S with linear switching cost where constant competitive ratios are achievable in 1-dimension.
  The dependency of our results on α compared to OCO-S (2) with linear switching cost where they are independent is because with OCO-S one can equivalently write the cost in (2) as $\hat{f}_t(.) + c(x(t), x(t-1))$, where $\hat{f}_t(.) = \frac{f_t(.)}{\alpha}$ that is still convex. Essentially, α can be absorbed within the convex function f_t revealed at time t itself in OCO-S (2) with linear switching cost.
- When the switching cost is quadratic, we propose a simple online algorithm, that at any time slot t chooses the number of servers to operate as $s(t) = \min\{\beta\sqrt{\frac{n(t)}{\alpha}}, n(t)\}$, where $\beta \geq 1$ (to be chosen).[2] We show that this algorithm has a competitive ratio of at most 20 for an appropriate choice of β, irrespective of the value of α.
  Thus, the linear switching cost case and quadratic switching cost case are fundamentally different in terms of α dependence. The main reason for this difference is that with the quadratic switching cost, OPT cannot avoid paying a comparable switching cost to the proposed algorithm, while in the linear switching cost case OPT can use its extra information about the future job arrivals and adjust its speed choices so as to pay an orderwise less switching cost than online algorithms when $\alpha > 1$.
  Compared to prior work on OCO-S for the quadratic switching cost [9], [13], [12] our analysis is novel, and uses the primal-dual technique to obtain the competitive ratio guarantee. In prior work on OCO-S, mainly a potential function approach is used that works well when there is no memory in the cost function.

II. System Model

Consider a discrete time system where each slot width is normalized to unity. Let the set of jobs that arrive over time be $\mathcal{J}$. For any job $j \in \mathcal{J}$, its arrival time is a_j while its size is $w_j \in \mathbb{Z}^+$, which is revealed on arrival. We will assume

[1] Static Regret$_T \equiv \sup_{\{f_t\}_{t=1}^T} \sup_{x^\star \in \mathcal{X}} \sum_{t=1}^T f_t(x_t) - \sum_{t=1}^T f_t(x^\star)$.

[2] When $\alpha < 1$, replace α with 1 in $s(t)$ definition.

identical job sizes for our analysis, i.e. $w_j = w_k,\ \forall j \neq k$, and let the job size be w. Without loss of generality, we let $w = 1$ from here onwards. We let all job arrivals and departures to happen at the start and end of a slot, respectively. From here on, time means a time slot. Following the classical work [18], we assume that there is an unlimited number of servers available, each with fixed speed of 1, and any job at any time instant can be processed by at most one server and each server can process at most one job at any time. Job j departs as soon as $w = 1$ amount of work has been completed for it. Let the departure time of job j be d_j. Then the flow time for job j is $f_j = d_j - a_j$, and the total flow time F is $\sum_{j\in\mathcal{J}} f_j$. Note that if $n(t)$ is the number of outstanding jobs at time t, then $F = \sum_t n(t)$.

Let the number of servers being used (called **active**) at time t be $s(t)$. Thus, at time t, $s(t)$ distinct jobs are getting processed in $s(t)$ servers at speed 1. Increasing the number of servers decreases the flow time, however, changing the number of active servers across time is costly and the switching cost is modelled as $c(s(t), s(t-1))$ that is generally an increasing function of $|s(t)-s(t-1)|$. Moreover, there is an energy cost needed to run $s(t)$ servers at time t that is modelled as $\theta \cdot s(t)$ for some $\theta > 0$.

Then the problem is to minimize the sum of the flow time, the switching cost, and the energy cost given by

$$\min_{s(t)}\ C(\alpha,\theta) = \sum_t n(t) + \alpha \sum_t c(s(t), s(t-1)) + \sum_t \theta \cdot s(t), \tag{3}$$

where α is the tradeoff factor between the importance given to the flow time and the switching cost. We consider two switching cost functions, linear $c(s(t), s(t-1)) = |s(t) - s(t-1)|$ and quadratic $c(s(t), s(t-1)) = (s(t) - s(t-1)^2$.

We consider the online setting where an online algorithm $\mathcal{A}$ at time t is only aware of the jobs that have arrived till time t, and makes its decisions about $s(t)$ to solve (3) using only that. Even $|\mathcal{J}|$ is not known to $\mathcal{A}$. To quantify the performance of $\mathcal{A}$, we use the metric of competitive ratio that is defined as follows.

We represent the optimal offline algorithm (that knows the entire job arrival sequence in advance) as OPT. For Problem (5), we will consider the metric of competitive ratio which for an algorithm $\mathcal{A}$ is defined as

$$\mu_{\mathcal{A}} = \max_{\sigma} \frac{C_{\mathcal{A}}(\alpha,\theta)}{C_{\mathsf{OPT}}(\alpha,\theta)}, \tag{4}$$

where σ is the input sequence consisting of jobs set $\mathcal{J}$. Since σ is arbitrary, we are not making any assumption on the job arrival times, or $|\mathcal{J}|$ the total number of jobs . The goal is to derive online algorithms with small competitive ratios.

Since all the jobs have to be completed, for any algorithm we have $\sum_t s(t) \geq \sum_{j\in\mathcal{J}} w_j$. Thus, it will turn out that it is sufficient to consider the cost

$$C(\alpha) = \sum_t n(t) + \alpha \sum_t c(s(t), s(t-1)), \tag{5}$$

to bound the competitive ratio (4).

A. Linear Switching Cost $c(s(t), s(t-1)) = |s(t) - s(t-1)|$

We first encounter the easy case of $\alpha \leq 1$ and then present the challenging one when $\alpha > 1$.

1) $\alpha \leq 1$: Consider an algorithm $\mathcal{A}_f$ that chooses $s(t) = n(t)$ for each time slot t.

Lemma 1: $\mathcal{A}_f$ is 2-competitive, and the competitive ratio is tight.

Essentially, when $\alpha \leq 1$, the cost of $\mathcal{A}_f$ is $\leq 2\sum_t n_{\mathcal{A}_f}(t)$, while $\sum_t n_{\mathcal{A}_f}(t)$ is a lower bound on the cost of OPT. All the missing proofs can be found in the Appendix.

2) $\alpha > 1$: This case is more interesting and challenging compared to $\alpha \leq 1$. With $\alpha > 1$, the most natural algorithm to minimize (3) inspired from prior work on OCO-S [18], [17], [3], [5], [1], [9], [4] is to choose $s(t)$ so as to balance the two instantaneous costs, flow-time $n(t)$ and switching cost $\alpha|s(t) - s(t-1)|$ which takes either of the two forms $s(t) = \frac{n(t)}{\alpha}$ or $|s(t) - s(t-1)| = \frac{n(t)}{\alpha}$. [3] We next show that for the considered problem algorithm $\mathcal{A}_{bal-1}$ that chooses $s(t) = \frac{n(t)}{\alpha}$ and $\mathcal{A}_{bal-2}$ that chooses $|s(t) - s(t-1)| = \frac{n(t)}{\alpha}$, both have unbounded competitive ratios.

Lemma 2: $\mathcal{A}_{bal-1}$ and $\mathcal{A}_{bal-2}$ have unbounded competitive ratios.

Thus, we need *non-balanced* algorithms. We present one next, and analyze its competitive ratio.

Algorithm $\mathcal{A}_L$: At time t, if the current active number of servers $s(t) > \frac{n(t)}{\alpha^{1/4}}$ do nothing. Otherwise, make $s(t) = \frac{n(t)}{\alpha^{1/4}}$, where $n(t)$ is the number of outstanding jobs at time t. Since $\alpha > 1$, $s(t) \leq n(t)$.

Lemma 3: $\mathcal{A}_L$ is at most $4\alpha^{1/4}$-competitive.

The proof uses a combination of two potential functions, one counting the number of active servers and other counting the outstanding amount of work between the algorithm and the OPT.

Proof: We need the following definition.

Definition 4: Work neutrality constraint: We define that an algorithm satisfies the work neutrality constraint if the amount of work done by time t cannot be more than the total amount of work that has arrived till then, i.e. $\sum_{t\leq t'} s(t) \leq \sum_{t\leq t'} w(t)$ for all t', where $w(t) = \sum_{j\in\mathcal{J}, a_j = t} w_j$ is the sum of sizes of jobs that arrive at time t. Essentially, servers are not allowed to idle in order to save on future switching costs.

To bound the competitive ratio, consider a variant of cost $C(\alpha)$ (5) where the switching cost is counted only when number of active servers is increased, and define it as

$$C^{\uparrow}_{\mathcal{A}}(\alpha) = \sum_t n(t) + \alpha \sum_t (s(t) - s(t-1))^+, \tag{6}$$

where $(x)^+ = \max\{0, x\}$.

$\mathcal{A}_L$ satisfies the work neutrality constraint, and since the switching cost is linear,

$$C_{\mathcal{A}_L}(\alpha) \leq 2C^{\uparrow}_{\mathcal{A}_L}(\alpha). \tag{7}$$

Next, we bound the cost (6) of $\mathcal{A}_L$ as a function of $C_{\mathsf{OPT}}(\alpha)$ by considering the following potential function

$$\Phi(t) = \Phi_1(t) + \Phi_2(t), \tag{8}$$

[3] Take ceiling if not an integer.

where

$$\Phi_1(t) = \alpha(-s(t) + s_o(t)), \tag{9}$$

where $s(t)$ and $s_o(t)$ are the number of active servers with $\mathcal{A}_L$ and OPT at time t, respectively, and

$$\Phi_2(t) = \alpha^{1/4}(W(t) - W_o(t)), \tag{10}$$

where $W(t) = n(t)$ and $W_o(t) = n_o(t)$ is the total work remaining with $\mathcal{A}_L$ and OPT at time t, and $n(t)$ and $n_o(t)$ are the number of outstanding jobs with $\mathcal{A}_L$ and OPT at time t, respectively.[4]

Clearly, $\Phi(0) = 0$ and $\Phi(\infty) = 0$. If at time t the number of active servers is increased by $\mathcal{A}_L$, i.e., if $s(t) > s(t-1)$

$$\begin{aligned}\Delta_1(t) &= \Phi_1(t) - \Phi_1(t-1),\\ &\leq \alpha(-s(t) + s(t-1)) + \alpha|s_o(t) - s_o(t-1)|, \end{aligned}\tag{11}$$

where $s(t) > s(t-1)$, and the first term is negative. Otherwise, if at time t the number of active servers is decreased by $\mathcal{A}_L$, i.e., if $s(t) < s(t-1)$

$$\begin{aligned}\Delta_1(t) &= \Phi_1(t) - \Phi_1(t-1),\\ &\leq \alpha(-s(t) + s(t-1)) + \alpha|s_o(t) - s_o(t-1)|, \end{aligned}\tag{12}$$

where $s(t) < s(t-1)$, and hence the first term is positive. When $s(t) = s(t-1)$,

$$\Delta_1(t) \leq \alpha|s_o(t) - s_o(t-1)|. \tag{13}$$

Moreover, for any t,

$$\begin{aligned}\Delta_2(t) &= \Phi_2(t) - \Phi_2(t-1),\\ &= \alpha^{1/4}(W(t) - W(t-1)) + \alpha^{1/4}(W_o(t-1) - W_o(t)),\\ &= \alpha^{1/4}(-s(t-1)) + \alpha^{1/4}(W_o(t-1) - W_o(t)),\\ &\overset{(a)}{\leq} -n(t-1) + \alpha^{1/4} n_o(t-1), \end{aligned}\tag{14}$$

where (a) follows by the choice of the number of active servers by $\mathcal{A}_L$ which ensures $s(t) \geq \frac{n(t)}{\alpha^{1/4}}$, while $(W_o(t) - W_o(t-1)) \leq n_o(t)$ follows from the fact that each job can be served by at most one server at any time and the speed of each server is unity.

Combining (11), (12), (13),(14), we get for $\mathcal{A}_L$ at any t,

$$\begin{aligned}&\underbrace{n(t) + \alpha(s(t) - s(t-1))^+}_{C^{\uparrow}_{\mathcal{A}_L}(\alpha,t)} + \Delta_1(t) + \Delta_2(t+1)\\ &\overset{(a)}{\leq} \alpha^{1/4} n_o(t) + \alpha|s_o(t) - s_o(t-1)|\\ &\quad + \mathbf{1}_{\downarrow}(t)\alpha(-s(t) + s(t-1)),\\ &\overset{(b)}{\leq} \alpha^{1/4} C_{\mathsf{OPT}}(\alpha, t) + \mathbf{1}_{\downarrow}(t)\alpha(-s(t) + s(t-1)), \end{aligned}\tag{15}$$

where in (a) $\mathbf{1}_{\downarrow}(t)$ is the indicator function that takes value 1 for slots t where $\mathcal{A}_L$ decreases its active number of servers and $(-s(t) + s(t-1)) > 0$ and is 0 otherwise, while in (b) $C_{\mathsf{OPT}}(\alpha, t)$ is the cost (5) of OPT at time t since $\alpha > 1$.

Lemma 5: For any algorithm, its flow time $F = \sum_t n(t) \geq \sum_{j\in\mathcal{J}} w_j$.

Follows trivially since each job can be processed by at most one server at any time and the speed of each server is unity, thus $d_j - a_j \geq w_j$.

[4]When $w \neq 1$, $W = wn(t)$ and $W = wn_o(t)$.

Lemma 6: For $\mathcal{A}_L$, $\sum_t (s(t) - s(t-1))^+ = \sum_t \mathbf{1}_{\downarrow}(t)(-s(t) + s(t-1))$ and $\sum_t (s(t) - s(t-1))^+ \leq \sum_{j\in\mathcal{J}} w_j$.

Both facts follow from the work neutrality constraint satisfaction for $\mathcal{A}_L$ and the linear switching cost.

Combining Lemma 5 and 6 from (6), we get that

$$\sum_t \mathbf{1}_{\downarrow}(t)\alpha(-s(t) + s(t-1)) \leq \frac{1}{2} C_{\mathcal{A}_L}. \tag{16}$$

Recall that $C^{\uparrow}_{\mathcal{A}_L}(\alpha) = \sum_t n(t) + \alpha(s(t) - s(t-1))^+$ and $\Phi(0) = 0$ and $\Phi(\infty) = 0$. Thus, summing (15) over all t, and using (18), we bound the cost (6) of $\mathcal{A}_L$, $C^{\uparrow}_{\mathcal{A}_L}(\alpha)$, as

$$C^{\uparrow}_{\mathcal{A}_L}(\alpha) \leq \alpha^{1/4} C_{\mathsf{OPT}}(\alpha) + \frac{1}{2} C^{\uparrow}_{\mathcal{A}_L}(\alpha). \tag{17}$$

Thus, (17) gives

$$C^{\uparrow}_{\mathcal{A}_L}(\alpha) \leq 2\alpha^{1/4} C_{\mathsf{OPT}}(\alpha). \tag{18}$$

Combining this with (7), we get

$$C_{\mathcal{A}_L}(\alpha) \leq 4\alpha^{1/4} C_{\mathsf{OPT}}(\alpha).$$

Recalling that $\mathcal{A}_L$ satisfies $\sum_t s(t) = \sum_{j\in\mathcal{J}} w_j$, while for OPT, $\sum_t s(t) \geq \sum_{j\in\mathcal{J}} w_j$ implies the result from Definition (4). □

Next, we show that the competitive ratio of any online algorithm belonging to the class $s(t) = \frac{n(t)}{\alpha^{\gamma}}$ for any $\gamma \geq 0$ is at least $\Omega(\alpha^{1/4})$. Note that choosing the number of active servers $s(t)$ at any time naturally constrains the switching cost for any algorithm as well.

Lemma 7: Consider an online algorithm $\mathcal{A}_{\gamma}$ that chooses $s(t) = \frac{n(t)}{\alpha^{\gamma}}$ for $\gamma \geq 0$. The competitive ratio of $\mathcal{A}_{\gamma}$ is $\Omega(\alpha^{1/4})$.

Discussion: When $\alpha \leq 1$, problem (3) is simpler than the OCO-S with linear switching cost, and a trivial algorithm is 2-competitive. In contrast when $\alpha > 1$, problem (3) is fundamentally different than OCO-S with linear switching cost. In particular, as pointed out before, a simple balancing algorithm is 3-competitive for OCO-S with linear switching cost [5], while as shown above the competitive ratio of any algorithm for problem (3) depends on α, and optimally grows as $\alpha^{1/4}$. Thus, even though there is great similarity between OCO-S and problem (3), the characterization is entirely different. This is essentially an artefact of the unbounded memory in the objective function with the considered problem.

B. Quadratic Switching Cost

In this subsection, we consider Problem (5) with quadratic switching cost, i.e.,

$$\min_{s(t)} \sum_t n(t) + \alpha \sum_t (s(t) - s(t-1))^2. \tag{19}$$

where $n(t) = \max\left\{\sum_{\tau\leq t} \text{arr}_{\tau} - \sum_{\tau\leq t-1} \text{dep}_{\tau}, 0\right\}$, where arr_{τ} and dep_{τ} are the number of jobs that arrive at the start of slot τ and depart at the end of slot τ, respectively.

Remark 8: With quadratic switching cost, we have to enforce that OPT **satisfies the work neutrality assumption**, defined in Definition 4. To see this, let servers be allowed to idle, i.e., violate the work neutrality assumption. Let $\alpha = 1$, and consider for example the following input, where at time $t = T$, T jobs of unit size arrive, and no jobs arrive before

and after time T. An offline algorithm O knowing this, will choose $s(t) = t$ for $t = 1, \ldots, T-1$, thus paying only a unit switching cost per unit time even when the switching cost is quadratic. At time T, when all the jobs arrive, O will choose $s(t) = T$, and finish all jobs in just one slot, and ramp down the speed as $s(T+t) = T-t$. Thus, O incurs a total switching cost of $2T$ and the flow time cost of T, resulting in a total cost of $3T$. In comparison, any online algorithm $\mathcal{A}$ cannot use $s(t) > 0$ until $t = T-1$ since otherwise the input can be such that no jobs arrive ever, making the competitive ratio of $\mathcal{A}$ arbitrary large. Thus, $\mathcal{A}$ can only choose $s(t) > 0$ for $t \geq T$, in which case its total cost is $\Omega(T^{3/2})$. Thus, if we do not make this critical modelling assumption, no algorithm can be competitive while choosing T large.

C. Algorithm

For solving (19), we consider the following algorithm, called $\mathcal{A}_Q$ from here on. **The choice of number of active servers** $s(t) = \min\{\beta\sqrt{\frac{n(t)}{\alpha}}, n(t)\}$, where $\beta \geq 1$ to be chosen later. If $s(t)$ is not an integer, take the ceiling. When $\alpha < 1$, replace α with 1 in $s(t)$ definition.

Scheduling: Since all jobs are of identical size, choose the $s(t)$ that have arrived the earliest and process them on $s(t)$ servers.

Remark 9: $\mathcal{A}_Q$ satisfies the work neutrality constraint.

Remark 10: As long as $\beta\sqrt{\frac{n(t)}{\alpha}} = \min\{\beta\sqrt{\frac{n(t)}{\alpha}}, n(t)\}$, $\mathcal{A}_Q$ chooses speed at time t such that the switching cost is at most β^2 times the number of remaining jobs at time t. Thus, $\mathcal{A}_Q$ tries to relate the two costs of (19).

Thus, an 'easy' upper bound on the total cost (19) of $\mathcal{A}_Q$ is as follows.

Lemma 11: The cost (19) of $\mathcal{A}_Q$ is at most $(1 + 2\beta^2)\sum_t n(t)$, where $n(t)$ is the number of remaining jobs at time t with the $\mathcal{A}_Q$.

It might appear that there is significant overcounting in the proof of Lemma 11 to upper bound the switching cost, however, one can construct input instances where the counting is in fact tight.

Following is the **main result** of this section.

Theorem 12: The competitive ratio of $\mathcal{A}_Q$ for solving (19) is at most 20.

The proof is rather long and is provided in the full version [25]. Here we only give a proof sketch as follows. Problem (19) can be equivalently written as

$$\min_{s(t)} \quad \sum_j \left(\sum_{t=a_j}^{d_j} s_j(t)\right) \frac{d_j - a_j}{w_j} + \alpha \sum_t (s(t) - s(t-1))^2$$
$$\sum_{t=a_j}^{d_j} s_j(t) \geq w_j, \ \forall\ j, \quad \sum_{\tau \leq t} s(\tau) \leq \sum_{\tau \leq t} w(\tau)\ \forall\ t,$$
$$s_j(t) \geq 0\ \forall\ t, j, \quad \sum_j s_j(t) = s(t)\ \forall\ t, \tag{20}$$

where $s_j(t) = 1$ if job j is being processed at time t and 0 otherwise, and $s(t)$ is the total number of active servers at time t, and the constraint $\sum_{t=a_j}^{d_j} s_j(t) \geq w_j$ implies that job j is eventually completed. The constraint $\sum_{\tau \leq t} s(\tau) \leq \sum_{\tau \leq t} w(\tau)\ \forall\ t$, corresponds to the work neutrality constraint and for brevity we denote it as $\mathsf{acc}(t) \leq D(t)$ hereafter, where $\mathsf{acc}(t) = \sum_{\tau \leq t} s(\tau)$, while $D(t) = \sum_{\tau \leq t} w(\tau)$.

The Lagrangian dual function associated with (20) is

$$\min_{s(t),\ d_j,\ \mathsf{acc}(t) \leq D(t)} \mathcal{L}(\lambda_j, \mu_j, \nu_t) = \min_{s(t),\ d_j,\ \mathsf{acc}(t) \leq D(t)} \sum_j \left(\sum_{t=a_j}^{d_j} s_j(t)\right) \frac{d_j - a_j}{w_j} + \alpha \sum_t (s(t) - s(t-1))^2$$
$$+ \sum_j \lambda_j \left(w_j - \sum_{t=a_j}^{d_j} s_j(t)\right) + \sum_j \mu_j s_j(t)$$
$$+ \sum_t \nu_t \left(\sum_j s_j(t) - s(t)\right), \tag{21}$$

where λ_j, μ_j, and ν_t are the dual variables corresponding to $\sum_{t=a_j}^{d_j} s_j(t) \geq w_j$, $s_j(t) \geq 0\ \forall\ t, j$, and $\sum_j s_j(t) = s(t)$, respectively, and **we have kept the work neutrality constraint as is in** (21)**.** Keeping the work neutrality constraint as it is is in fact critical for the analysis. Instead if it is included inside the Lagrangian, that results in a trivial lower bound that is not useful. This feature makes the analysis sufficiently novel compared to usual application of primal-dual scheme for similar problems.

Choosing the dual variable λ_j Given $\mathcal{A}_Q$, choose λ_j such that $\lambda_j w_j = \Delta_j F_{\mathcal{A}_Q}$, where $\Delta_j F_{\mathcal{A}_Q}$ is the increase in the total flow time of the $\mathcal{A}_Q$ because of arrival of job j with size w_j at time a_j, disregarding jobs arriving after time a_j.

For this choice of λ_j, we obtain the following important structural result for $\mathcal{A}_Q$.

Lemma 13: $\lambda_j - \left(\frac{t - a_j}{w_j}\right) \leq \frac{3}{\beta}(\alpha n(t))^{1/2}$ for all t and $\beta \geq 1$, where $n(t)$ is the number of remaining jobs with $\mathcal{A}_Q$ at t.

Combining Lemma 13 and invoking work neutrality, we get the following lower bound on the Lagrangian in (21)

$$\geq \sum_t n(t) \left(1 - \frac{9}{4\beta^2}\right), \tag{22}$$

where $n(t)$ is the number of remaining jobs with $\mathcal{A}_Q$ at time t.

From the weak duality, the primal value of (19) is at least as much as the value of any dual feasible solution. Thus, the competitive ratio of any online algorithm $\mathcal{A}$ is at most the ratio of an upper bound on the cost of $\mathcal{A}$ and a lower bound on the dual feasible solution. Hence combining (22) and Lemma 11 and the fact that both $\mathcal{A}_Q$ and OPT satisfy the work neutrality constraint, we get that the competitive ratio of $\mathcal{A}_Q$ is at most $\mu_{\mathcal{A}_Q} \leq \frac{1+2\beta^2}{\left(\frac{4\beta^2-9}{4\beta^2}\right)}$. For $\beta^2 = 3$, we get $\mu_{\mathcal{A}_Q} \leq 7 \times 4 = 28$, while optimizing over β, we get that the optimal $\beta = 2.177$ and the competitive ratio is 19.95. A practical value to use is $\beta = 2$. Full proof of Theorem 12 is provided in the full version [25].

Discussion: Theorem 12 shows that $\mathcal{A}_Q$ that tries to balance the flow time and switching cost of (19) has a constant competitive ratio. This balancing idea is also integral to algorithms analyzed for OCO-S with quadratic switching cost problem,

however, the analysis of $\mathcal{A}_Q$ is entirely different compared to OCO-S.

The usual technique to analyze OCO-S with quadratic switching cost problem is to consider a potential function $\Phi_1(x(t), x^o(t)) = ||x(t) - x^o(t)||^2$, where $x(t)$ and $x^o(t)$ are the actions chosen by an algorithm and the OPT at time t, respectively, and critically use the property that the optimal choice to minimize the cost $f_t(.)$ in (2) is $x_t^\star$ that depends only on f_t. Since this is not the case with (19), we deviate from the potential function approach and consider the primal-dual scheme.

The main advantage of the primal-dual scheme is that there is no need to characterize the structure or any other property of the OPT directly, which is anyway not available for (19). By exploiting the monotonicity property of $\mathcal{A}_Q$ (an algorithm is defined to be **monotone** if $n^{\mathcal{I}}(t) \leq n^{\mathcal{I}'}(t)$, where $n^{\mathcal{I}}(t)$ is the number of remaining jobs at time t with job arrival instance $\mathcal{I}$ and $\mathcal{I} \subseteq \mathcal{I}'$) and the fact that any new job does not preempt an existing job (which is true since all job sizes are identical) for $\mathcal{A}_Q$, we derive an upper bound on the primal cost and a lower bound on the dual cost with $\mathcal{A}_Q$, which together with the weak duality implies the competitive ratio bound.

The analysis we use is similar to [19] that considered a flow-time +energy minimization problem but without any switching costs or discrete number of active servers constraint or the work neutrality constraint. Compared to usual primal-dual technique analysis, e.g. [19], the most challenging new part is the enforcement of the work neutrality constraint as described in Remark 8. As far as we know primal-dual analysis has not been applied for solving the OCO-S with quadratic switching cost problem (2) or its variants, and we believe it is going to be quite useful for other OCO-S problems as well. It is important to note that our analysis does not work for arbitrary objective convex functions $f_t's$ as described in OCO-S with quadratic switching cost, thus our work critically exploits the specific objective function of (19).

III. Stochastic setting

We also consider a stochastic variant of the flow time plus switching cost minimization problem analysed in the preceding sections. Our goal as before is to design algorithms that are constant competitive (relative to the optimal online algorithm). Interestingly, we find that the nature of constant competitive algorithms in the stochastic setting (with Poisson arrivals) differs from those we established as competitive in the worst case setting (where burst arrivals are possible). Finally, we also consider a single server speed scaling variant of our model in the stochastic setting, where the aggregate service rate is not constrained by the number of outstanding jobs, and show that this relaxation results in a significant cost reduction in heavy traffic. Remarkably, we show that in this case, overall cost that grows $o(\lambda)$ is attainable (where λ is job-arrival rate) with a policy that operates at a single speed, but only after letting the buffer build to a certain threshold. For lack of space, the precise model and the theoretical results can be found in the full version [25].

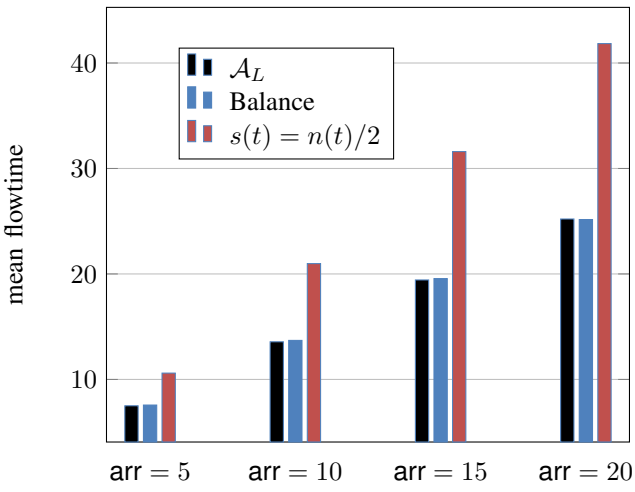

Fig. 1. Comparison of mean flow time with different algorithms as a function of mean arrival rate per slot with linear switching cost with $\alpha = 1$.

IV. Numerical results

In this section, we present simulation results for the sum of the flow time (per job)+ total switching cost divided by the time horizon. For all simulations, with arrival rate arr, we generate arr jobs on average per slot and then distribute them over the time horizon arbitrarily. We take the time horizon large, and compute the mean flow time (summed across jobs) + total switching cost normalized by the length of the time horizon.

We start with linear switching cost case with $\alpha = 1$ and $\alpha = 2$ for arrival rates $5, 10, 15, 20$. For $\alpha = 1$, in Fig. 1, we compare the performance of the proposed algorithm $s(t) = n(t)$ with an algorithm that chooses $s(t) = n(t)/2$ and the *Balance* algorithm $|s(t) - s(t-1)| = n(t)$. Next, we consider $\alpha = 2$ in Fig. 2 and compare the performance of the proposed algorithm with the two *Balance* algorithms. Fig. 1, and Fig. 2 reveal that the performance of the proposed algorithm is better than other algorithms.

The quadratic switching case is considered next, where we consider the case of $\alpha = 1, 2$ for arrival rates $5, 10, 15, 20$ for different choice of parameter β of the proposed algorithm and *Balance* algorithm in Figs. 3 and 4. These numerical results are interesting, where at times a larger arrival rate leads to smaller sum of the flow time (per job)+ total switching cost divided by the time horizon, since when there are larger number of jobs, the number of active servers is chosen large early enough and does not need to changed often enough over the time horizon, leading to small switching cost together with small flow time.

From Figs. 3, and 4, it also appears that the performance of the proposed algorithm improves as β is increased from 1 to 4, and the performance with $\beta = 4$ is similar to that of the *Balance* algorithm. Recall that in our theoretical guarantee, the optimal choice of β was close to 2. From Fig. 3 and Fig. 4 it appears that as the arrival rate increases the performance of the proposed algorithm degrades compared to the *Balance* algorithm.

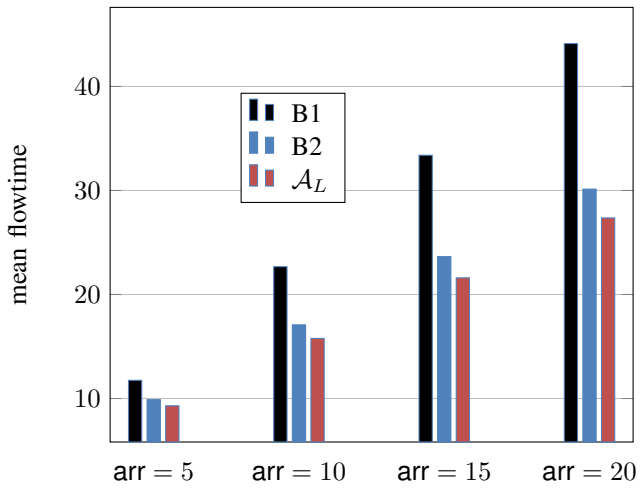

Fig. 2. Comparison of mean flow time with different algorithms as a function of mean arrival rate per slot with linear switching cost with $\alpha = 2$, where B1 stands for Balance algorithm that uses $s(t) = n(t)/\alpha$, while B2 stands for Balance algorithm that uses $|s(t) - s(t-1)| = n(t)$.

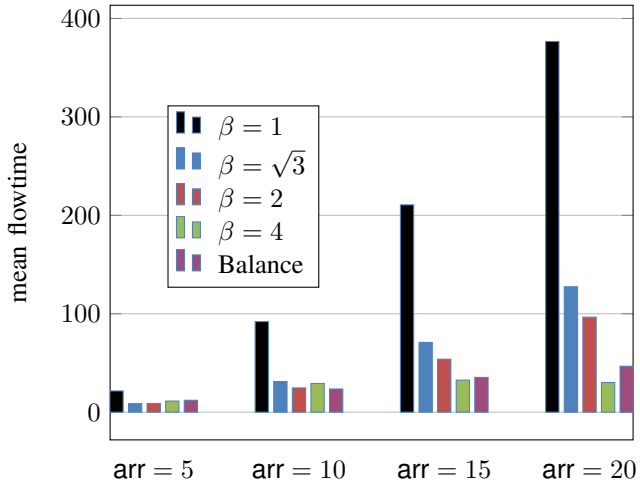

Fig. 3. Comparison of mean flow time with the proposed algorithm with different values of β and Balance $\alpha(s(t) - s(t-1))^2 = n(t)$ for quadratic switching cost with $\alpha = 1$ as a function of mean arrival rate per slot.

V. Conclusions

In this paper, we have broadened the horizon of OCO-S problem by incorporating two important aspects: i) the objective function has unbounded memory, i.e., at any time depends on all the actions taken so far, and ii) the considered objective function motivated by capacity provisioning where the queuing dynamics interact with server switching cost is non-convex. Prior work on OCO indicated that the regret or the competitive ratio increases linearly in the memory size, however, using the specific structure of the considered problem, we avoided the negative results, and derived algorithms with memory independent competitive ratios.

References

[1] S. Albers and J. Quedenfeld. Optimal algorithms for right-sizing data centers - extended version. *CoRR*, abs/1807.05112, 2018.

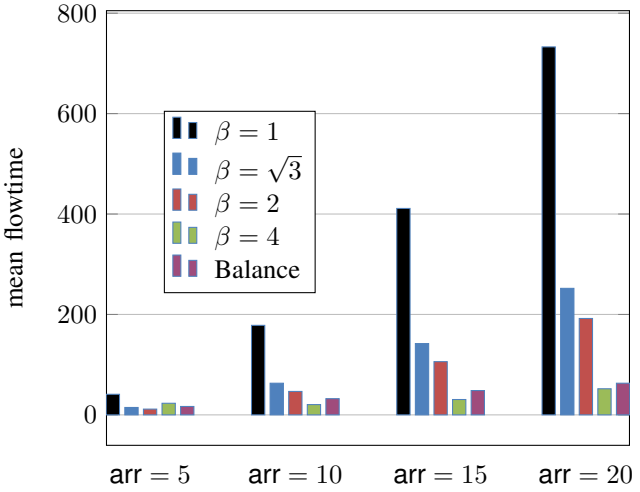

Fig. 4. Comparison of mean flow time with the proposed algorithm with different values of β and Balance $\alpha(s(t) - s(t-1))^2 = n(t)$ for quadratic switching cost with $\alpha = 2$ as a function of mean arrival rate per slot.

[2] O. Anava, E. Hazan, and S. Mannor. Online convex optimization against adversaries with memory and application to statistical arbitrage. *arXiv preprint arXiv:1302.6937*, 2013.

[3] L. Andrew, S. Barman, K. Ligett, M. Lin, A. Meyerson, A. Roytman, and A. Wierman. A tale of two metrics: Simultaneous bounds on competitiveness and regret. In *Conference on Learning Theory*, pages 741–763, 2013.

[4] C. Argue, A. Gupta, and G. Guruganesh. Dimension-free bounds for chasing convex functions. In *Conference on Learning Theory*, pages 219–241. PMLR, 2020.

[5] N. Bansal, A. Gupta, R. Krishnaswamy, K. Pruhs, K. Schewior, and C. Stein. A 2-competitive algorithm for online convex optimization with switching costs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[6] N. Bhuyan, D. Mukherjee, and A. Wierman. Best of both worlds: Stochastic and adversarial convex function chasing, 2023.

[7] N. Chen, A. Agarwal, A. Wierman, S. Barman, and L. L. Andrew. Online convex optimization using predictions. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 191–204, 2015.

[8] N. Chen, J. Comden, Z. Liu, A. Gandhi, and A. Wierman. Using predictions in online optimization: Looking forward with an eye on the past. *ACM SIGMETRICS Performance Evaluation Review*, 44(1):193–206, 2016.

[9] N. Chen, G. Goel, and A. Wierman. Smoothed online convex optimization in high dimensions via online balanced descent. In *Conference On Learning Theory*, pages 1574–1594. PMLR, 2018.

[10] N. Chen, G. Goel, and A. Wierman. Smoothed online convex optimization in high dimensions via online balanced descent. In *Conference On Learning Theory*, pages 1574–1594. PMLR, 2018.

[11] N. Christianson, T. Handina, and A. Wierman. Chasing convex bodies and functions with black-box advice. In *Conference on Learning Theory*, pages 867–908. PMLR, 2022.

[12] G. Goel, Y. Lin, H. Sun, and A. Wierman. Beyond online balanced descent: An optimal algorithm for smoothed online optimization. *Adv. Neural Inf. Process. Syst.*, 32, 2019.

[13] G. Goel and A. Wierman. An online algorithm for smoothed regression and LQR control. *Proceedings of Machine Learning Research*, 89:2504–2513, 2019.

[14] R. Kumar, S. Dean, and R. Kleinberg. Online convex optimization with unbounded memory. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] A. Lechowicz, N. Christianson, B. Sun, N. Bashir, M. Hajiesmaili, A. Wierman, and P. Shenoy. Chasing convex functions with long-term constraints, 2024.

[16] Y. Li, G. Qu, and N. Li. Using predictions in online optimization with switching costs: A fast algorithm and a fundamental limit. In *2018*

Annual American Control Conference (ACC), pages 3008–3013. IEEE, 2018.

[17] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking*, 21(5):1378–1391, 2012.

[18] M. Lin, A. Wierman, A. Roytman, A. Meyerson, and L. L. Andrew. Online optimization with switching cost. *ACM SIGMETRICS Performance Evaluation Review*, 40(3):98–100, 2012.

[19] K. T. Nguyen. Lagrangian duality in online scheduling with resource augmentation and speed scaling. In *Algorithms–ESA 2013: 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings 21*, pages 755–766. Springer, 2013.

[20] F. Orabona. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019.

[21] D. Rutten, N. Christianson, D. Mukherjee, and A. Wierman. Smoothed online optimization with unreliable predictions. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 7(1):1–36, 2023.

[22] S. Senapati and R. Vaze. Online convex optimization with switching cost and delayed gradients. *Perform. Eval.*, 162(C), Feb 2024.

[23] G. Shi, Y. Lin, S.-J. Chung, Y. Yue, and A. Wierman. Online optimization with memory and competitive control. *Advances in Neural Information Processing Systems*, 33:20636–20647, 2020.

[24] R. Vaze. *Online Algorithms*. Cambridge University Press, 2023.

[25] R. Vaze and J. Nair. Capacity provisioning motivated online non-convex optimization problem with memory and switching cost, 2024.

[26] L. Zhang, W. Jiang, S. Lu, and T. Yang. Revisiting smoothed online learning. *CoRR*, abs/2102.06933, 2021.

VI. APPENDICES

A. Proof of Lemma 1

Proof: With $\mathcal{A}_f$ each outstanding job is under service from one server at any point of time, which is the best possible from any job's point of view in minimizing its flow time. Thus, the flow time of any job is just $\sum_t n(t)$, where $n(t)$ is the number of jobs present in the system when each job is processed at unit speed as soon as it arrives. Moreover, the total switching cost of $\mathcal{A}$ at time t is at most $\alpha n(t)$. Therefore, the total cost of $\mathcal{A}$ is at most $(1+\alpha)\sum_t n(t)$. From the above discussion we get that OPT's flow time $\sum_t n_o(t) \geq \sum_t n(t)$, since any job can be processed by at most one with unit speed at any time. Thus, $(1+\alpha)\sum_t n(t) \leq (1+\alpha)\sum_t n_o(t)$, and we get that the competitive ratio of $\mathcal{A}$ is at most $1+\alpha$. Given $\alpha \leq 1$, the competitive ratio is at most 2.

We next provide an instance where the competitive ratio of $\mathcal{A}_f$ is close to 2. Consider an input where at times $2, 4, \ldots, 2k$ for some $k \in \mathbb{Z}^+$, x (even) number of jobs arrive, each with size 1. The proposed algorithm will choose $s(t) = x$ for time slots $2, 4, \ldots, 2k$ and $s(t) = 0$ otherwise, resulting in a total flow time of $x \cdot k$ and a total switching cost of $2kx$. In contrast, consider another algorithm $\mathcal{B}$ that processes $x/2$ jobs in each time slot starting from time slot 2. $\mathcal{B}$ pays a switching cost of $x/2$ only at time 2 and at time $2k+1$ making the total switching cost just x. Moreover, the flow time of algorithm $\mathcal{B}$ is $k(x/2 + 2x/2) = 1.5kx$. Since OPT is at least as good as $\mathcal{B}$, the competitive ratio of the proposed algorithm is at least $\frac{3xk}{1.5xk+x} \approx 2$ choosing k large. □

B. Proof of Lemma 2

Proof: We begin with $\mathcal{A}_{bal-1}$ Let the total number of jobs be N and all the N jobs arrive at time 0 and let $\alpha = \mathsf{N}$. Then the algorithm $\mathcal{A}_{bal-1}$ keeps $s(t) = 1$ always, and its flow time is $\Omega(\mathsf{N}^2)$. An alternate algorithm $\mathcal{B}$ that uses $s(t) = \frac{\mathsf{N}}{\sqrt{\alpha}}$ at time 0 and until all jobs are finished has a switching cost of $2\sqrt{\alpha}\mathsf{N} = O(\mathsf{N}^{3/2})$ and a flow time of $\sqrt{\alpha}\mathsf{N} = O(\mathsf{N}^{3/2})$. Note that $\sum_t s(t) = \sum_{j\in\mathcal{J}} w_j = \mathsf{N}$ for $\mathcal{A}_{bal-1}$, while $\sum_t s(t) \geq \sum_{j\in\mathcal{J}} w_j = \mathsf{N}$ for any algorithm. Since OPT is at least as good as $\mathcal{B}$, we get that the competitive ratio (4) of $\mathcal{A}_{bal-1}$ is $\Omega(\mathsf{N}^{1/2})$. Choosing N large we get the result.

Next, we consider the algorithm $\mathcal{A}_{bal-2}$ with the input where N jobs arrive at time 0 and $\alpha = \mathsf{N}^2$. Algorithm $\mathcal{A}_{bal-2}$ has a switching cost of $\Omega(\mathsf{N}^{2.5})$, while an alternate algorithm $\mathcal{B}$ that uses $s(t) = \frac{\mathsf{N}}{\sqrt{\alpha}}$ at time 0 and until all jobs are finished has a switching cost of $2\sqrt{\alpha}\mathsf{N} = O(\mathsf{N}^2)$ and a flow time of $\sqrt{\alpha}\mathsf{N} = O(\mathsf{N}^2)$. Once again we invoke the fact that $\sum_t s(t) = \sum_{j\in\mathcal{J}} w_j = \mathsf{N}$ for $\mathcal{A}_{bal-1}$, while $\sum_t s(t) \geq \sum_{j\in\mathcal{J}} w_j = \mathsf{N}$ for any algorithm. Since OPT is at least as good as $\mathcal{B}$, we get that the competitive ratio (4) of $\mathcal{A}_{bal-2}$ is $\Omega(\mathsf{N}^{1/2})$. Choosing N large we get the result. □

VII. PROOF OF LEMMA 7

Proof: Consider two input sequences σ_1 and σ_2. With σ_1, N jobs arrive at time 1 and no more jobs arrive thereafter, while with σ_2, N jobs arrive at each time $t = 1, \ldots, T$.

Let $\mathcal{A}_\gamma$ be an online algorithm that chooses $s(t) = \frac{n(t)}{\alpha^\gamma}$ for $\gamma \geq 0$. If $\gamma < 1/4$ then the input is σ_1, else if $\gamma \geq \frac{1}{4}$ then the input is σ_2,.

With σ_1, let an algorithm $\mathcal{B}$ choose $s(t) = \frac{n(t)}{\sqrt{\alpha}}$. Then the total cost of $\mathcal{B}$, $C_\mathcal{B}(\sigma_1) \leq 3\sqrt{\alpha}\mathsf{N}$. Since OPT is as good as $\mathcal{B}$, $C_{\mathsf{OPT}}(\sigma_1) \leq 3\sqrt{\alpha}\mathsf{N}$ $\mathcal{A}_\gamma$ on the other hand has a switching cost of $\alpha^{1-\gamma}\mathsf{N}$ and thus, for $\gamma < 1/4$

$$\frac{C_{\mathcal{A}_\gamma}(\sigma_1)}{C_{\mathsf{OPT}}(\sigma_1)} \geq \Omega(\alpha^{\frac{1}{2}-\gamma}). \tag{23}$$

Next, consider σ_2, where N jobs arrive at each time slot for $t = 1, \ldots, T$, we will pick T large later. Consider an algorithm $\mathcal{B}$ that chooses $s(1) = \mathsf{N}$ and $s(t) = s(1)$ until time T. The cost of $\mathcal{B}$ is at most $\alpha\mathsf{N} + \mathsf{N}T$. Given that OPT is at least as good as $\mathcal{B}$, $C_{\mathsf{OPT}}(\sigma_2) \leq \alpha\mathsf{N} + \mathsf{N}T$.

For $\mathcal{A}_\gamma$, that chooses $s(t) = \frac{n(t)}{\alpha^\gamma}$, the number of outstanding jobs at time $n(t)$ satisfies the recursion $n(t) = n(t-1) + \mathsf{N} - \frac{n(t-1)+\mathsf{N}}{\alpha^\gamma}$. Simple algebra[5] reveals that for any t, $\tau < t \leq T$ where τ is $o(T)$, the number of jobs remaining $n(t) = (\alpha^\gamma - 1)\mathsf{N}$. Thus, the flow time of $\mathcal{A}_\gamma$ is at least $\mathsf{N}(\alpha^\gamma - 1)T$. Thus, $C_{\mathcal{A}_\gamma} \geq \Omega(\mathsf{N}\alpha^\gamma T)$. Choosing T large enough compared to α, for large enough α, we get that

$$\frac{C_{\mathcal{A}_\gamma}(\sigma_2)}{C_{\mathsf{OPT}}(\sigma_2)} \geq \Omega(\alpha^\gamma). \tag{24}$$

Combining (23) and (24), we get that the competitive ratio of $\mathcal{A}_\gamma$ is $\geq \max\{\alpha^\gamma, \alpha^{\frac{1}{2}-\gamma}\}$. Thus, to minimize its competitive ratio $\mathcal{A}_\gamma$ should choose $\gamma = \frac{1}{4}$, and the resulting competitive ratio lower bound is $\Omega(\alpha^{1/4})$. □

[5]Fixed point of recursion $n(t) = n(t-1) + \mathsf{N} - \frac{n(t-1)+\mathsf{N}}{\alpha^\gamma}$ is $n(t) = (\alpha^\gamma - 1)\mathsf{N}$.