Revisiting Online Learning Meta-Algorithm From the Perspective of Weighted Alpha-Fair Allocation

Zhiyuan Wang, Jiancheng Ye, *Senior Member, IEEE*, and John C. S. Lui, *Fellow, IEEE*

***Abstract*—Multiplicative Weight Update (MWU) is an online learning meta-algorithm that has been widely used in sequential decision-making problems. The key idea of MWU is to maintain a distribution as the selection probability allocation to the alternative actions. MWU has been independently rediscovered and investigated in many studies. This paper will revisit MWU from the perspective of fair resource allocation. Specifically, we focus on a well-known fairness concept called weighted alpha-fairness, which was introduced by Mo and Walrand in network bandwidth allocation. We first unveil that MWU inherently adopts the weighted alpha-fair probability allocation in each slot. The fairness level equals to the inverse of the step-size parameter in MWU, and the weight assigned to each action decays exponentially in relation to its accumulated cost. This insight suggests that the use of appropriate weighted-alpha fair allocations for action-taking probability can indeed perform as well as the offline optimal allocation in the long-term. Motivated by this insight, we further investigate whether one could solve more general online resource allocation problems (not limited to the one addressed by MWU) by carefully constructing the weighted-alpha fair allocation for each time slot. To this end, we propose a weighted alpha-fair online allocation framework that carefully constructs the merit-based weights and the increasing fairness levels. Under our proposed framework, the allocation outcome in each time slot satisfies the weighted alpha-fairness principal, and the allocation outcomes also perform asymptotically as well as the offline optimal outcome. We demonstrate how this framework functions in two networking applications. In size-based traffic scheduling, our framework enables the network switches to prioritize short flows and avoid flow starvation without the prior flow size information. In file caching, our framework outperforms several state-of-the-art caching policies up to 21% in terms of cache-hit-ratio.**

***Index Terms*—Online learning, network resource allocation, multiplicative weight update (MWU), weighted alpha-fairness.**

I. Introduction

THIS paper presents the inherent connection between a meta-algorithm of online learning and a fairness concept in network resource allocation. The meta-algorithm is called Multiplicative Weight Update (MWU). It has been widely used in many online learning problems [2]. The fairness concept is called weighted alpha-fairness, introduced by Mo and Walrand in [3]. It is a fairness measurement that is often adopted in network utility maximization problems. Overall, this meta-algorithm and fairness concept were independently proposed in last century, and have been independently investigated for decades. In this paper, we would like to present their inherent connection as well as the usage of the connection.

A. Multiplicative Weighted Update

The key idea of Multiplicative Weighted Update (MWU) is to maintain a distribution (i.e., selection probability) over a set of N alternative actions according to their weights to be updated (according to their performance) in a multiplicative manner. Despite its simplicity, this idea carries significant impact in many different areas. In game theory, an algorithm similar to MWU was proposed in early 1950s. In online learning theory, to our knowledge, the earliest version of MWU is the Winnow algorithm proposed by Littlestone in [4], [5]. Later on, some versions of MWU were proposed in the classic problem called prediction with expert advices. For example, Littlestone and Warmuth in [6] proposed the weighted majority algorithm. Later on, Freund and Schapire in [7] proposed the Hedge algorithm. Due to the widespread use of MWU, Arora et al. [2] regard it as a meta-algorithm and introduce various disparate algorithms as straightforward instantiations of this meta-algorithm.

The meta-algorithm MWU has been independently rediscovered in many studies. In this paper, we revisit MWU through the lens of fair resource allocation, exploring its implications in a new context. Our foundational premise is that the distribution orchestrated by MWU can be conceptualized as an allocation of the selection probability to alternative actions This perspective leads us to focus on the following key question that arises from this understanding.

Question 1: Does the selection probability distribution used in MWU meet certain criteria for fairness?

B. Fairness of Resource Allocation

Resource allocation is a fundamental research problem in many computer and communication network systems, e.g., bandwidth allocation (e.g., [8], [9], [10]), caching capability allocation (e.g., [11]), ad space allocation (e.g., [12]), to name a few. This problem usually encompasses the interactions

Received 14 January 2025; revised 4 August 2025; accepted 17 October 2025; approved by IEEE TRANSACTIONS ON NETWORKING Editor L. Ying. Date of publication 10 November 2025; date of current version 5 January 2026. This work was supported in part by the National Natural Science Foundation of China under Grant U24B20128 and Grant 62202021. The work of John C. S. Lui was supported in part by the Research Grants Council (RGC) Grant GRF-14202923. An earlier version of this paper was presented at the ACM MobiHoc 2022 [DOI: 10.1145/3492866.3549724]. *(Corresponding author: Jiancheng Ye.)*

Zhiyuan Wang is with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China (e-mail: zhiyuanwang@buaa.edu.cn).

Jiancheng Ye is with the School of Computer Science and Engineering, Faculty of Innovation Engineering, Macau University of Science and Technology, Macau, China (e-mail: jcye@must.edu.mo).

John C. S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China (e-mail: cslui@cse.cuhk.edu.hk).

This article has supplementary downloadable material available at https://doi.org/10.1109/TON.2025.3626194, provided by the authors.

Digital Object Identifier 10.1109/TON.2025.3626194

between the system operator and the resource requesters (or the network nodes) in the network system. The payoff of the system operator is positively related to the system efficiency, which the system operator seeks to maximize. Each resource requester wants to obtain more resource, thus the requester population usually cares about the fairness of the resource allocation outcome. Fairness is an abstract metric for equity and has a wide range of definitions. Mo and Walrand in [3] introduce the *weighted alpha-fairness*, which is defined based on the weighted alpha-fair utility function. As the name suggests, this utility function is parameterized by a weight vector $\boldsymbol{w} = (w_1, w_2, \ldots, w_N)$ and a scaler α. A feasible allocation of the resource to the N nodes is considered to be weighted alpha-fair if it maximizes the aggregated weighted alpha-fair utility. This principle has been extensively applied in the realm of network resource allocation (e.g., bandwidth allocation [8], [9], [10]). It generalizes several fairness definitions such as the proportional fairness (i.e., the Nash bargaining solution [13]) and the max-min fairness (i.e., the Kalai-Smorodinsky bargaining solution [14]). In a more recent study, Lan et al. in [15] elucidate the implications of a larger alpha value in achieving a more fair allocation outcome.

As we will show in Section III, the selection probability allocation adopted in MWU satisfies weighted alpha-fairness for certain weight vectors and alpha values. This insight motivates us to dive deeper into the relationship between weighted alpha-fairness and the traditional online learning paradigm (i.e., exploration-exploitation tradeoff). This leads to the following key question.

Question 2: What is the relationship between weighted alpha-fair allocations and the online learning paradigm?

C. Weighted Alpha-Fair Online Allocation

Given the connection between MWU and weighted alpha-fairness, we are interested in whether the weighted alpha-fair allocations are capable of solving some more general online resource allocation problems (not limited to the one addressed by MWU). In resource allocation problem, the system efficiency is usually defined by the system operator to characterize the system performance of its own interest (e.g., operation expenditure). Hence the performance of resource allocation often has different mathematical forms. In many networking systems (e.g., [16], [17], [18], [19], [20], [21], [22], [23], [24], [25]), system efficiency may even involve unknown parameters due to the changing environment (e.g., dynamics topology [26]). In this case, the system operator is not able to accurately anticipate the system performance (i.e., the efficiency metric) until the unknown parameters are completely revealed. We briefly introduce some typical networking applications that fall into this category: edge caching capacity allocation for files with unknown request rate or file popularity (e.g., [16]), traceback resource allocation for received malicious packets (e.g., [17]), wireless spectrum allocation for multiple IoT sensors with unknown sensing data amount (e.g., [18]), display space allocation for multiple advertisers with unknown click rates (e.g. [19]), and job dispatching to multiple geo-distributed servers with unknown energy expense (e.g., [20]). Most studies adopt the online convex optimization framework (e.g., [27], [28]) without considering the allocation fairness. If we could establish the connection between weighted alpha-fairness and the learning rationale, then it is interesting to further explore whether proper weighted alpha-fair allocations ensure optimality in online resource allocation problems. This is our third key question.[1]

Question 3: How to perform as well as the offline optimal allocation via weighted alpha-fairness in general online allocation problems?

D. Main Results & Key Contributions

This paper aims to elucidate the intrinsic relationship between the online learning meta-algorithm MWU and weighted alpha-fairness. Our research is structured into two main components. First, we analyze the distribution maintained by MWU as a selection probability allocation among the alternative actions and assess its fairness in resource distribution. This allows us to figure out the connection between exploration-exploitation tradeoff and weighted alpha-fairness. Second, based on the findings above, we extend our focus to more general online resource allocation problems, and investigate how to utilize instantaneous weighted alpha-fair allocations to perform as well as the offline optimal allocation in the long-term. Our main results and key contributions are as follows:

- *Understanding MWU from the Perspective of Weighted Alpha-Fairness:* Our investigation on the selection probability allocation within MWU reveals that MWU inadvertently adopts the weighted alpha-fair probability allocation in each slot. The fairness level equals to the inverse of the step-size parameter in MWU, and the weight assigned to each action decays exponentially in relation to its accumulated cost. This discovery suggests that the use of appropriate weighted-alpha fair allocations for action-taking probability can indeed perform as well as the offline optimal allocation in the online setting.
- *Connection between Weighted Alpha-Fairness and Learning Rationale:* We unveil the inherent connection between the concept of weighted alpha-fairness and the universal learning rationale. Specifically, the weight vector corresponds to the exploitation based on previous feedbacks, whereas the fairness level dictates the extent of exploration within a dynamically changing environment.
- *Design of Weighted Alpha-Fair Online Allocations:* Given the established connection between weighted alpha-fairness and learning rationale, we propose a novel weighted alpha-fair online allocation framework. This framework is designed by carefully constructing the merit-based weights and the increasingly adjusting the fairness level. Under this framework, the allocation in each slot adheres to the weighted alpha-fair principal, and the allocations also perform asymptotically as well as the offline optimal outcome.

[1]In this paper, the the interpretation of "perform as well as the offline optimal allocation" does not imply that the regret matches the lower bound.

- *Case Studies:* We demonstrate how the proposed framework functions in two networking applications. In network traffic scheduling, our framework enables the network switches to prioritize the short flows and avoid the flow starvation without requiring any prior knowledge on the flow size distribution. In the domain of file caching, our framework outperforms several state-of-the-art caching policies up to 21% in terms of the cache-hit-ratio under the typical file request models.

We would like to clarify that the algorithmic framework in this paper builds upon well-established online optimization techniques. The key contribution is the recognition of the new interpretation of MWU's behavior as implicitly following weighted alpha-fair allocation. This reinterpretation provides a unifying perspective that was previously unexplored in the literature. The conceptual connection offers new insights into why these heuristics work well in practice.

The remainder of this paper is as follows. Section II introduces related works. Section III revisits MWU from the perspective of fair resource allocation. Section IV and Section V present the design of weighted alpha-fair online allocation framework for general online allocation problems (without and with constraints, respectively). Section VI provides a case study. We conclude this paper in Section VII.

II. Literature Review

A. Fairness in Resource Allocation

Previous studies on utility-guided allocation focus on the fairness-efficiency tradeoff of various resource allocation problems, which are usually static in terms of the system efficiency and the minimal guarantee. Typical applications include the network bandwidth allocation (e.g., [8]) and path selection (e.g., [29], [30]). Proportional fairness and max-min fairness are two classic fairness criteria, which comes from the Nash bargaining solution [13] and Kalai-Smorodinsky bargaining solution [14], respectively. The *weighted alpha-fairness*, introduced by Mo and Walrand in [3], is a prevalent measurement and generalizes proportional fairness (i.e., $\alpha = 1$) and max-min fairness (i.e., $\alpha \to \infty$). Furthermore, some studies on caching capacity allocation (e.g., [11]) and ad space allocation (e.g., [12]) also follow the utility maximization framework. These studies model the requests based on Poisson process, and define the utility as a function of the stationary hit probability. More recently, Lan et al. in [15] extend this scheme and demonstrate what it actually means for a larger alpha value to be more fair. Jeo-Wong et al. in [31] extend this scheme to the scenario with multi-dimensional resource.

Our study in this paper does not aim to change or improve the above fairness concept. Instead, we leverage the weighted alpha-fairness to understand the selection probability allocation in the online learning meta-algorithm MWU.

B. Online Resource Allocation

There have been many works (e.g., [16], [17], [18], [19], [20], [21], [22], [23], [24], [25]) on online resource allocation problems, where the system performance is revealed sequentially after adopting the allocation decisions. These studies aim to maximize the efficiency metric asymptotically (neglecting the fairness issue) in online allocation problems. Most of them follow the online learning/optimization theory (e.g., [27], [28]), and typical applications include the file caching problems (e.g., [16], [17]), the wireless channel access problems (e.g., [18]), the advertisement display space allocation problems (e.g. [19]), and the job dispatching problems (e.g., [20]). These studies focus primarily on improving the system efficiency (or equivalently attain a sub-linear regret), yet overlook the fairness issue. More recently, Li et al. in [21] propose a combinatorial bandit framework with fairness constraints. Nevertheless, the fairness constraints essentially represent the minimum guarantee requirement, which is also captured in other studies (e.g., [22], [23], [24], [25]) and this paper.

To sum up, previous studies on online resource allocation merely focus on the optimality. Different from previous studies, our work in this paper aims to ensure the optimality in online resource allocations by enforcing proper weighted alpha-fair allocations instantaneously. Therefore, the goal of the allocation policy design in our work substantially differs from that in the aforementioned works.

C. This Paper

Our study in this paper starts with understanding the inherent connection between online learning meta-algorithm MWU and weighted alpha-fairness. The new finding guides us to propose a weighted alpha-fair online allocation framework for general online resource allocation problems. Our proposed framework is capable of ensuring the allocation efficiency and the minimum guarantee in the long-term by adopting the instantaneous weighted alpha-fair allocations.

III. Understanding MWU Through the Lens of Weighted Alpha-Fairness

In this section, we try to understand MWU from the perspective of weighted alpha-fairness. We will demonstrate that MWU essentially induces weighted-alpha fair allocations for the selection probability among the alternative actions. We first provide some preliminary knowledge on MWU and weighted alpha-fairness in Section III-A and Section III-B, respectively. We then reveal their inherent relationship in Section III-C.

A. Problem Setting and Procedure of MWU

We provide some preliminary knowledge on multiplicative weight update (MWU) in the following.

1) Problem Setting: MWU is an online learning algorithm for the decision-maker who needs to take sequential actions given a fixed action set $\mathcal{N} = \{1, 2, \ldots, N\}$ during multiple time slots $t \in \{1, 2, 3, \ldots\}$. Let $c_t[n] \in [0, 1]$ denote the cost of taking action $n \in \mathcal{N}$ in time slot t. Due to the environment uncertainty, the decision-maker does not know the cost vector $\boldsymbol{c}_t = (c_t[n] : \forall n \in \mathcal{N})$ until the end of slot t. The event sequence in each time slot is as follows.

- At the beginning of slot t, the decision-maker selects an action, denoted by $n_t \in \mathcal{N}$.
- At the end of slot t, the environment is revealed and the decision-maker observes $\boldsymbol{c}_t = (c_t[n] : \forall n \in \mathcal{N})$.

Let $\mathfrak{A}$ denote a generic policy that generates the sequential actions $\{n_t : \forall t = 1, 2, 3, \ldots\}$ for the decision-maker. The performance of policy $\mathfrak{A}$ is usually evaluated based on the following cumulative regret up to time T:

$$Reg_T(\mathfrak{A}) \triangleq \mathbb{E}\left[\sum_{t=1}^{T} c_t[n_t]\right] - \min_{n\in\mathcal{N}} \sum_{t=1}^{T} c_t[n], \tag{1}$$

where the expectation is taken over the randomness in the policy $\mathfrak{A}$. If the policy $\mathfrak{A}$ achieves a sub-linear regret, i.e., $Reg_T(\mathfrak{A}) \sim o(T)$, then the policy $\mathfrak{A}$ is considered to perform as well as the offline optimal outcome [27], [28].

2) Procedure of MWU: Algorithm 1 summarizes the procedure of MWU. Roughly speaking, MWU maintains a selection probability distribution over the N actions. The selection probability allocated to action $n \in \mathcal{N}$ in slot t is $p_t[n] \in [0, 1]$. Accordingly, $\boldsymbol{p}_t = (p_t[n] : \forall n \in \mathcal{N})$ represents the selection probability distribution in slot t. Moreover, the selection probability $p_t[n]$ will be negatively related to the cumulative cost of action n across previous slots. Initially, the selection probability distribution is uniform, i.e., Line 1. Later on, each time slot consists of the following two phases:

- Lines 3-4: At the beginning of each slot t, MWU randomly draws a number $n_t \in \mathcal{N}$ according to the selection probability distribution $\boldsymbol{p}_t$, and adopts the action n_t.
- Lines 5-7: At the end of slot t, the decision-maker observes the cost vector $\boldsymbol{c}_t$ and calculates the cumulative cost of each action. MWU then updates the selection probability $\boldsymbol{p}_{t+1}$ based on the cumulative cost. Note that this updating step involves step-size parameter η_t, which is non-increasing in t.

So far, we have introduced the problem setting and major procedure of MWU. Theorem 1 summarizes the theoretic performance achieved by MWU. The proof could be found in previous studies (e.g., Chapter 4.2 of [27] and Chapter 1.3.1 of [28]), thus is omitted here.

Theorem 1 (Performance of MWU) For any non-increasing sequence $\{\eta_t : \forall t = 1, 2, \ldots\}$, the regret incurred by MWU in Algorithm 1 satisfies

$$Reg_T(\mathsf{MWU}) \leq \frac{\ln(N)}{\eta_T} + \sum_{t=1}^{T} \frac{\eta_t}{8}. \tag{2}$$

Let $\eta_t = \sqrt{4\ln(N)/t}$, then we have

$$Reg_T(\mathsf{MWU}) \leq \sqrt{T\ln(N)}. \tag{3}$$

Theorem 1 shows that MWU achieves a sub-linear regret. This result is fundamentally linked to the selection probability allocation $\{\boldsymbol{p}_t : \forall t \geq 1\}$ and the step-size sequence $\{\eta_t : \forall t \geq 1\}$. Next we first provide some preliminary knowledge on weighted-alpha fairness in Section III-B. We then elucidate $\{\boldsymbol{p}_t : \forall t \geq 1\}$ and $\{\eta_t : \forall t \geq 1\}$ from the perspective weighted alpha-fairness in Section III-C.

Algorithm 1 Multiplicative Weight Update (MWU)

1 **Initialize** $\boldsymbol{p}_1 = (p_1[n] = \frac{1}{N} : \forall n \in \mathcal{N})$ and a non-increasing sequence $\{\eta_t : t = 1, 2, 3, ...\}$
2 **for** $t = 1$ **to** T **do**
  `/* At the beginning of this slot */`
3 **Draw** a random number $n_t \in \mathcal{N}$ according to the selection probability $\boldsymbol{p}_t = (p_t[n], \forall n \in \mathcal{N})$
4 **Adopt** the action n_t in this slot
  `/* At the end of this slot */`
5 **Observe** the cost vector $\boldsymbol{c}_t = (c_t[n] : \forall n \in \mathcal{N})$
6 **Calculate** the cumulative cost $(c_{1:t}[n] : \forall n \in \mathcal{N})$ of each action according to

$$c_{1:t}[n] = \sum_{\tau=1}^{t} c_\tau[n], \quad \forall n \in \mathcal{N}.$$

7 **Calculate** the selection probability

$$p_{t+1}[n] = \frac{\exp\left(-\eta_t c_{1:t}[n]\right)}{\sum_{i=1}^{N} \exp\left(-\eta_t c_{1:t}[i]\right)}, \quad \forall n \in \mathcal{N}.$$

B. Weighted Alpha-Fair Allocation

Consider the allocation of the total resource A to a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of N nodes. Let $\boldsymbol{x} = (x_n : \forall n \in \mathcal{N})$ denote an allocation outcome, which is chosen from the feasible set

$$\mathcal{X} \triangleq \left(\boldsymbol{x} \in \mathbb{R}_+^N : \sum_{n=1}^{N} x_n = A\right). \tag{4}$$

There are many criteria to evaluate the fairness of the allocation outcome $\boldsymbol{x}$. A general class of fairness is called utility fairness, which is defined based on a concave utility function. A feasible allocation that maximizes the utility function is said to satisfy the corresponding utility fairness. Weighted alpha-fairness is a special case of utility fairness when the utility function is the following weighted alpha-fair utility [3]

$$U(\boldsymbol{x}; \boldsymbol{w}, \alpha) \triangleq \begin{cases} \sum_{n=1}^{N} w_n \log(x_n), & \alpha = 1, \\ \sum_{n=1}^{N} \frac{w_n x_n^{1-\alpha}}{1-\alpha}, & \alpha \geq 0, \alpha \neq 1, \end{cases} \tag{5}$$

where $\boldsymbol{x} \in \mathcal{X}$ is a feasible allocation decision. Note that $U(\boldsymbol{x}; \boldsymbol{w}, \alpha)$ is parameterized by a weight vector $\boldsymbol{w} = (w_n \geq 0 : \forall n \in \mathcal{N})$ and a fairness level $\alpha \geq 0$. The feasible allocation that maximizes the weighted alpha-fair utility $U(\boldsymbol{x}; \boldsymbol{w}, \alpha)$ satisfies the corresponding $(\boldsymbol{w}, \alpha)$-fairness [3], which is presented in Definition 1.

Definition 1 (Weighted Alpha-Fairness [3]) Given the weight vector $\boldsymbol{w} \in \mathbb{R}_+^N$ and the fairness level $\alpha \geq 0$, the following allocation outcome $\hat{\boldsymbol{x}}$ is $(\boldsymbol{w}, \alpha)$-fair.

$$\hat{\boldsymbol{x}} \triangleq \arg\max_{\boldsymbol{x}\in\mathcal{X}} \quad U(\boldsymbol{x}; \boldsymbol{w}, \alpha). \tag{6}$$

The weighted alpha-fairness generalizes the proportional fairness when $\alpha = 1$ and the max-min fairness when $\alpha \to \infty$. It is usually believed that a larger α value corresponds to a more fair allocation outcome [15]. Note that (6) is a convex

optimization problem, and the Karush-Kuhn-Tucker (KKT) conditions enable us to derive $\hat{\boldsymbol{x}}$ in a closed-form as shown in Proposition 1.

Proposition 1: When $\alpha > 0$, the $(\boldsymbol{w}, \alpha)$-fair allocation of the total resource A is given by

$$\hat{x}_n = \frac{w_n^{1/\alpha}}{\sum_{i=1}^{N} w_i^{1/\alpha}} \cdot A, \quad \forall n \in \mathcal{N}. \tag{7}$$

C. Relation Between MWU and Weighted Alpha-Fairness

Now we try to understand MWU through the lens of fair resource allocation. Our foundational premise is that the distribution orchestrated by MWU can be conceptualized as an allocation of the selection probability to the alternative actions. Recall that MWU maintains a selection probability distribution $\{\boldsymbol{p}_t : \forall t \geq 1\}$ over the N actions according to

$$p_t[n] = \frac{\exp\left(-\eta_{t-1} c_{1:t-1}[n]\right)}{\sum_{i=1}^{N} \exp\left(-\eta_{t-1} c_{1:t-1}[i]\right)}, \quad \forall n \in \mathcal{N}. \tag{8}$$

That is, MWU needs to allocate a unit probability (i.e., the resource) to the N actions. Since it can be viewed as a resource allocation problem, then a natural question is whether the probability allocation $\boldsymbol{p}_t$ is fair and what kind of fairness criteria $\boldsymbol{p}_t$ satisfies. Proposition 2 provides the answer from the angle of weighted alpha-fairness.

Proposition 2: The selection probability allocation $\boldsymbol{p}_t = (p_t[n] : \forall n \in \mathcal{N})$ adopted by MWU is weighted alpha-fair with the parameters $(\bar{\boldsymbol{w}}_t, \bar{\alpha}_t)$, which is given by

$$\begin{aligned} \bar{w}_t[n] &= \exp\big(-c_{1:t-1}[n]\big), \quad \forall n \in \mathcal{N}, \\ \bar{\alpha}_t &= \frac{1}{\eta_{t-1}}, \end{aligned} \tag{9}$$

where $c_{1:t-1}[n]$ is the cumulative cost incurred by action $n \in \mathcal{N}$, and η_{t-1} is the step-size.

Proposition 2 indicates that MWU inadvertently adopts a weighted alpha-fair probability allocation $\boldsymbol{p}_t$ in each slot t. Specifically, the fairness level equals to the inverse of the step-size parameter in MWU, and the weight assigned to each action decays exponentially in relation to its accumulated cost. Overall, Proposition 2 provides a new understanding on how to perform as well as the offline optimal outcome and *why it can achieve* in online learning problems.

How: Recall that the traditional understanding on online learning is to address the dilemma between exploitation and exploration. It is widely acknowledged that striking the right balance between exploitation and exploration is the core of achieving asymptotic optimality. Proposition 2 provides another understanding on how to achieve asymptotic optimality in online learning problems:

The use of appropriate weighted-alpha fair allocations for action-taking probability can indeed guarantee asymptotic optimality in online learning problems.

Why: We elucidate why the appropriate weighted-alpha fair allocations for action-taking probability can guarantee asymptotic optimality. First, the weight $\bar{w}_{t,n}$ is positively related to the merit achieved by action n in previous time slots, and the probability allocation amount $p_{t,n}$ is positively related to the weight $\bar{w}_{t,n}$ of action n. Hence the weight vectors $\boldsymbol{w}_t$ control the exploitation in MWU. Second, the fairness level $\bar{\alpha}_t$ controls the magnitude of the weight vector's impact. In the case of $\bar{\alpha}_t \to 0$, all the selection probability goes to the action that incurs the least cost up to now. In the case of $\bar{\alpha}_t \to \infty$, the selection probability tends to be equally shared among the alternative actions. That is, a larger $\bar{\alpha}_t$ means a stronger fairness requirement, which reduces the weight vector's impact and leads to a stronger exploration in MWU. The two aspects provide with valuable hints on how the weighted alpha-fair allocations manipulate the exploitation-exploration tradeoff.[2]

The relation built in Section III shows that MWU's behaviors in each slot satisfy weighted alpha-fairness. This leads to new insights into why these heuristics work well in online learning problems. Based on the insights, we would like to further investigate whether one could solve some more general online allocation problems by properly customizing the weighted-alpha fair allocation in each slot. The answer is yes. To provide a simple demonstration, we focus on unconstrained online resource allocation in Section IV, and then go one step further to constrained online resource allocation in Section V. We would like to clarify that the proposed framework may not lead to substantial performance improvement, but enables us to explicitly leverage fairness-aware tuning in online allocation problems, where MWU has been used but fairness was not previously an explicit consideration.

TABLE I
KEY NOTATIONS

Symbol	Physical Meaning
$\mathcal{N}$	The set of N nodes requesting the resource.
A	Amount of resource to be allocated in each slot.
$x_{t,n}$	Amount of resource allocated to node n in slot t.
$f_t(\cdot)$	The loss function in slot t.
$g_{t,m}(\cdot)$	The m-th minimal guarantee requirement in slot t.
G	The partial derivative bound defined in (22).
$U(\boldsymbol{x}; \boldsymbol{w}, \alpha)$	Weighted alpha-fair utility function.
$\mathcal{U}$	Weighted alpha-fair online allocation framework.
$\boldsymbol{w}_t$	Weight vector at slot t under our framework $\mathcal{U}$
α_t	Fairness level of slot t under our framework $\mathcal{U}$
$\hat{\boldsymbol{x}}_t$	The $(\boldsymbol{w}_t, \alpha_t)$-fair allocation outcome.
β_t	The parameter defined in (27b).
δ_t	The parameter defined in (27a).
$Reg_T(\cdot)$	The efficiency loss in the regret form up to slot T.
$Vio_T(\cdot)$	The minimal guarantee violation up to slot T.

IV. ONLINE ALLOCATION

This section focuses on the general online resource allocation problems, and presents how to utilize instantaneous weighted alpha-fair allocations to achieve asymptotic optimality in the long-term. We start with introducing the general problem setting of online allocation in Section IV-A. We then present the design of the weighted alpha-fair online allocation framework in Section IV-B. Table I shows the key notations.

[2]Similar results have also been discussed by Dehghan et al. in [11] under the context of caching policy design. However, [11] does not consider the connection between the weighted alpha-fairness and the exploration-exploitation tradeoff, which is the key insight of our work in this paper.

A. Online Allocation Problem

Consider a networking system where the resource (or workload) needs to be allocated to some requesters in an online manner across the operation horizon $\mathcal{T} = \{1, 2, \ldots, T\}$ with T slots. Specifically, the system operator will allocate a total of $A \geq 0$ resource to a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of N requesters. Due to the changing environment, the system operator is not able to anticipate the system performance in advance, thus has to dynamically adjust its allocation decisions in an online manner. There are two phases in each time slot: At the beginning of each time slot t, the system operator will allocate a total of A resource to the N requesters. At the end of time slot t, the environment is revealed, thus the operator perceives the feedbacks regarding the system performance. Next, we characterize the allocation decision and the allocation performance for the general online allocation problem.

1) Allocation Decision: We let $x_{t,n} \geq 0$ denote the amount of allocation to the requester $n \in \mathcal{N}$ in slot t. Accordingly, the operator's allocation decision in slot t is given by

$$\boldsymbol{x}_t \triangleq (x_{t,n} \in \mathbb{R}_+ : \forall n \in \mathcal{N}). \tag{10}$$

Given the total resource amount A, the allocation decision $\boldsymbol{x}_t$ is supposed to be chosen from the set feasible $\mathcal{X}$. Note that the equality condition in (4) does not reduce the generality in this problem. Essentially, it can explicitly capture both the network workload allocation and the network resource allocation. For workload allocation (e.g., job dispatching to geo-distributed servers), the front-end job router will dispatch all the arriving jobs to the N server clusters [20]. For resource allocation (e.g., caching capacity allocation), a larger x_n means more allocated resource, thus will not make node n worse-off [11].

2) Allocation Performance: We model the efficiency of a networking system based on the general loss functions. Such a loss function may represent the negative cache-hit-ratio in caching capacity allocation or the energy expenditure in job dispatching problems. Mathematically, we let $f_t(\cdot)$ denote the loss function in slot t, and make two-fold elaborations on it. First, the system operator does not know the loss function $f_t(\cdot)$ until the end of slot t due to the changing environment (e.g., unknown file requests or electricity prices). Hence the system operator has to determine $\boldsymbol{x}_t$ based only on the information revealed in previous slots $\{1, 2, \ldots, t-1\}$. Second, the loss functions $\{f_t(\cdot) : \forall t \in \mathcal{T}\}$ may vary over time. In this work, we suppose that the loss functions are convex and time-varying possibly in a non-i.i.d. or adversarial manner (due to byzantine nodes in networks [32].

3) Offline Benchmark: Given the above problem setting, we define the offline optimal allocation decision as follows.

Definition 2: Given the revealed loss functions $\{f_t(\cdot) : \forall t \in \mathcal{T}\}$, the offline optimal allocation decision is given by

$$\boldsymbol{x}^\star \triangleq \arg\min_{\boldsymbol{x} \in \mathcal{X}} \sum_{t=1}^{T} f_t(\boldsymbol{x}). \tag{11}$$

The total loss incurred by $\boldsymbol{x}^\star$ is the benchmark that we want to achieve in the online allocation policy design. Next, we present the problem formulation of the weighted alpha-fair online allocation framework.

4) Problem Formulation: We aim to design a weighted alpha-fair online allocation framework, denoted by $\mathcal{U}$, for the general online resource allocation problem. Such a framework is built upon the classic weighted alpha-fair utility (5), and should be able to tackle the environment uncertainty. To achieve this goal, we will customize the weight vectors $\{\boldsymbol{w}_t : \forall t \in \mathcal{T}\}$ and the fairness levels $\{\alpha_t : \forall t \in \mathcal{T}\}$. The sequence $\{(\boldsymbol{w}_t, \alpha_t) : \forall t \in \mathcal{T}\}$ is the "*design space*" of the weighted alpha-fair online allocation framework $\mathcal{U}$. That is, we intend to design a framework $\mathcal{U}$ such that the *instantaneously* $(\boldsymbol{w}_t, \alpha_t)$-fair allocations, i.e.,

$$\hat{\boldsymbol{x}}_t = \arg\max_{\boldsymbol{x} \in \mathcal{X}} U(\boldsymbol{x}; \boldsymbol{w}_t, \alpha_t), \quad \forall t \in \mathcal{T}, \tag{12}$$

are able to perform as well as the offline optimal outcome *in the long-term*. For performance analysis, we follow previous studies (e.g., [16], [17], [18], [19], [20], [21], [22], [23], [24]) and measure the *efficiency loss* of our proposed framework $\mathcal{U}$ based on the cumulative regret:

$$Reg_T(\mathcal{U}) \triangleq \sum_{t=1}^{T} \mathbb{E}\left[f_t(\hat{\boldsymbol{x}}_t)\right] - \sum_{t=1}^{T} \mathbb{E}\left[f_t(\boldsymbol{x}^\star)\right], \tag{13}$$

where $\boldsymbol{x}^\star$ is the offline benchmark in Definition 2, and $\{\hat{\boldsymbol{x}}_t : t \in \mathcal{T}\}$ denote the decisions generated according to (12) in our framework $\mathcal{U}$. Note that if $Reg_T(\mathcal{U})$ is sub-linear in T, then the adopted online allocations $\{\hat{\boldsymbol{x}}_t : \forall t \in \mathcal{T}\}$ are instantaneously $(\boldsymbol{w}_t, \alpha_t)$-fair, and can perform as well as the offline optimal outcome in the long-term.

B. Weighted Alpha-Fair Online Allocation Framework

Next we present the weighted alpha-fair online allocation framework $\mathcal{U}$ by appropriately devising the weight vectors $\{\boldsymbol{w}_t : \forall t \in \mathcal{T}\}$ and the fairness levels $\{\alpha_t : \forall t \in \mathcal{T}\}$ such that the efficiency loss $Reg_T(\mathcal{U})$ are sub-linear in T. Our study in this paper relies on the following assumption regarding the convexity and the bounded partial derivative, which are common conditions made in related works (e.g., [27], [28]).

Assumption 1: In each slot t, the loss function $f_t(\boldsymbol{x})$ are convex in $\boldsymbol{x} \in \mathcal{X}$, and have a bounded partial derivative, i.e.,

$$\left|\frac{\partial f_t(\boldsymbol{x})}{\partial x_n}\right| \leq G, \quad \forall t \in \mathcal{T}. \tag{14}$$

We will introduce the major components of our proposed weighted alpha-fair online allocation framework $\mathcal{U}$.

1) Weighted Alpha-Fair Allocation: In each slot t, the weighted alpha-fair online allocation framework $\mathcal{U}$ will adopt the $(\boldsymbol{w}_t, \alpha_t)$-fair allocation $\hat{\boldsymbol{x}}_t$ defined in (12). Recall that $\hat{\boldsymbol{x}}_t$ has a closed-form expression in Proposition 1, thus this step exhibits a low computational complexity. The subsequent two components will specify the weight vectors $\{\boldsymbol{w}_t : \forall t \in \mathcal{T}\}$ and the fairness levels $\{\alpha_t : \forall t \in \mathcal{T}\}$, respectively,

2) Merit-Based Weight: The weighted alpha-fair online allocation framework $\mathcal{U}$ will adopt the merit-based weight:

$$w_{t+1,n} = w_{t,n} \cdot \exp\left(-\frac{\partial f_t(\hat{\boldsymbol{x}}_t)}{\partial x_n}\right), \quad \forall n \in \mathcal{N}, \tag{15}$$

where $\hat{\boldsymbol{x}}_t$ is the adopted allocation decision in slot t. Note that the updating rule in (15) is based on an exponential function, which is inspired by Proposition 2.

3) Increasing Fairness Level: The weighted alpha-fair online allocation framework $\mathcal{U}$ will adopt the following fairness level in each slot t

$$\alpha_t = \frac{t^{\epsilon} G}{2\sqrt{\ln(N)}}, \quad \forall t \in \mathcal{T}, \tag{16}$$

where $\epsilon \in (0,1)$ is determined by the system operator. Note that α_t is increasing in t for any $\epsilon \in (0,1)$. Moreover, $\epsilon \in (0,1)$ is a tunable parameter in this framework, which actually determines the tradeoff between the fairness level and the efficiency loss (in the regret form).

Algorithm 2 summarizes the weighted alpha-fair online allocation framework $\mathcal{U}$. It first specifies the parameter $\epsilon \in (0,1)$ in Line 2. We then initialize the weight vector $\boldsymbol{w}_1$ as the all-one vector. Each time slot consists of two phases:

- *Lines 4-5:* At the beginning of slot t, we calculate the fairness level α_t according to (16), and then adopt the $(\boldsymbol{w}_t, \alpha_t)$-fair allocation $\hat{\boldsymbol{x}}_t$.
- *Lines 6-7:* At the end of slot t, the loss function $f_t(\cdot)$ is revealed. We then calculate the weight vector $\boldsymbol{w}_{t+1}$ for the next slot according to (15).

So far, we have introduced weighted alpha-fair online allocation framework $\mathcal{U}$. Theorem 2 presents the theoretic performance. The proof is given in appendix (see supplementary material).

Theorem 2: For any $\epsilon \in (0,1)$, the weighted alpha-fair online allocation framework $\mathcal{U}$ in Algorithm 2 achieves the following efficiency loss (i.e., regret)

$$Reg_T(\mathcal{U}) \leq \left(\frac{T^{1-\epsilon}}{1-\epsilon} + \frac{T^{\epsilon}}{2}\right) AG\sqrt{\ln(N)}. \tag{17}$$

Algorithm 2 Weighted Alpha-Fair Online Allocation $\mathcal{U}$

Input : A, N, and G
Output: Allocation decisions $\hat{\boldsymbol{x}}_t$ for each slot t.
1 **Determine** $\epsilon \in (0,1)$
2 **Initialize** $\boldsymbol{w}_1 = \mathbf{1}_N$
3 **for** $t = 1$ **to** T **do**
/* At the beginning of this slot */
4 **Calculate** the fairness level α_t according to (16).
5 **Adopt** the $(\boldsymbol{w}_t, \alpha_t)$-fair allocation decision $\hat{\boldsymbol{x}}_t$.
/* At the end of this slot */
6 **Observe** the revealed loss function $f_t(\cdot)$
7 **Update** the weight vector $\boldsymbol{w}_{t+1}$ according to (15).

Theorem 2 shows that the *weighted alpha-fair* allocations $\{\hat{\boldsymbol{x}}_t : \forall t \in \mathcal{T}\}$ performs asymptotically as well as the offline efficient/optimal benchmark $\boldsymbol{x}^{\star}$ in Definition 2. Moreover, the parameter $\epsilon \in (0,1)$ controls the tradeoff between the allocation fairness level $O(T^{\epsilon})$ and the efficiency loss $O\left(\frac{T^{1-\epsilon}}{1-\epsilon} + \frac{T^{\epsilon}}{\epsilon}\right)$.

V. Online Allocation With Minimum Guarantee

This section will go one step further and focuses on the general online resource allocation problems with constraints. Similarly, we will introduce how to ensure sub-linear regret via customizing weighted alpha-fair online allocations.

A. Online Allocation Problems With Constraints

1) Problem Setting: Consider the online allocation problem setting in Section IV-A. Besides the loss function $\{f_t(\cdot) : \forall t \in \mathcal{T}\}$, we suppose that the system operator also needs to ensure some other critical minimum guarantee requirements in real-world networking systems. Such a requirement may represent the time-average link utilization constraints or the queuing stability constraints, which could be violated instantaneously but should be satisfied in the long-term. In general, the minimum guarantee requirements or constraints could be time-varying in real-world networking applications. We follow the previous studies (e.g., [22], [23], [24]) and suppose that these constraints vary over time in an i.i.d. manner (i.e., stochastic constraints).[3] Mathematically, we consider a set $\mathcal{M} = \{1, 2, \ldots, M\}$ of M stochastic constraints as follows

$$\{g_{t,m}(\boldsymbol{x}) \leq 0 : \forall m \in \mathcal{M}\}, \tag{18}$$

where the function $g_{t,m}(\boldsymbol{x}) \triangleq \tilde{g}_m(\boldsymbol{x}; \theta_t)$ is convex in $\boldsymbol{x} \in \mathcal{X}$ and uniquely determined by the random variable θ_t. Moreover, $\{\theta_t : \forall t \in \mathcal{T}\}$ are i.i.d. random variables that capture the environment uncertainty. In practice, the system operator does not possess the probability distribution of the random variables $\{\theta_t : \forall t \in \mathcal{T}\}$, and is not able to predict their realizations. That is, the system operator can only observe the realized constraints (18) at the end of each slot t, and updates the future allocation decisions.

2) Offline Benchmark: Given the above problem setting, we define the offline optimal/efficient allocation as follows.

Definition 3: Given the revealed loss functions $\{f_t(\cdot) : \forall t \in \mathcal{T}\}$ and the distribution of random variables $\{\theta_t : \forall t \in \mathcal{T}\}$, the offline optimal/efficient allocation decision is

$$\begin{aligned} \boldsymbol{x}^{\star} \triangleq \arg\min_{\boldsymbol{x} \in \mathcal{X}} \; & \sum_{t=1}^{T} \mathbb{E}\left[f_t(\boldsymbol{x})\right] \\ s.t. \; & \mathbb{E}\left[\tilde{g}_m(\boldsymbol{x}; \theta_t)\right] \leq 0, \quad \forall m \in \mathcal{M}, \end{aligned} \tag{19}$$

where the expectation is taken over the random variable θ_t.

3) Problem Formulation: We aim to design a weighted alpha-fair online allocation framework, denoted by $\mathcal{U}$, for the general online resource allocation problem with constraints. We will design the weight $\{\boldsymbol{w}_t : \forall t \in \mathcal{T}\}$ and the fairness level $\{\alpha_t : \forall t \in \mathcal{T}\}$ such that the $(\boldsymbol{w}_t, \alpha_t)$-fair allocations

$$\hat{\boldsymbol{x}}_t = \arg\max_{\boldsymbol{x} \in \mathcal{X}} U(\boldsymbol{x}; \boldsymbol{w}_t, \alpha_t), \quad \forall t \in \mathcal{T}, \tag{20}$$

perform asymptotically as well as the offline benchmark $\boldsymbol{x}^{\star}$ in Definition 3. On the one hand, we will measure the efficiency loss based on regret defined in (13). On the other hand, we will evaluate the cumulative violation of the minimum guarantee requirement according to

$$Vio_T(\mathcal{U}) \triangleq \sum_{m=1}^{M} \mathbb{E}\left[\sum_{t=1}^{T} g_{t,m}(\hat{\boldsymbol{x}}_t)\right]^{+}, \tag{21}$$

[3]Mannor et al. in [33] show that it is impossible to achieve the sub-linear regret and constraint violation if both the loss functions and constraints vary arbitrarily. Hence many studies (e.g., [22], [23], [24]) focus on the case with the non-i.i.d. loss functions and the i.i.d. constraints.

where $[\cdot]^+ = \max(\cdot, 0)$, and the expectation is taken with respect to the random variables $\{\theta_t : \forall t \in \mathcal{T}\}$. Our study in this section relies on the following assumption regarding the convexity and the bounded partial derivative, which are common conditions made in related works (e.g., [27], [28]).

Assumption 2: In each slot t, the loss function $f_t(\boldsymbol{x})$ and the constraint functions $\{g_{t,m}(\boldsymbol{x}) : \forall m \in \mathcal{M}\}$ are convex in $\boldsymbol{x} \in \mathcal{X}$, and have a bounded partial derivative, i.e.,

$$\left|\frac{\partial f_t(\boldsymbol{x})}{\partial x_n}\right| \leq G, \ \left|\frac{\partial g_{t,m}(\boldsymbol{x})}{\partial x_n}\right| \leq G, \ \forall n \in \mathcal{N}, m \in \mathcal{M}. \quad (22)$$

B. Weighted Alpha-Fair Online Allocation With Constraints

Now we introduce how to devise the weighted alpha-fair online allocation framework $\mathcal{U}$. Roughly speaking, the merit-based weight vectors should simultaneously reflects the system efficiency and the minimum guarantee. Hence we follow the previous studies (e.g., [22], [23], [24], [25]), and consider the following Lagrangian function in each slot t

$$\mathcal{L}_t(\boldsymbol{x}_t, \boldsymbol{\mu}_t) \triangleq f_t(\boldsymbol{x}_t) + \sum_{m=1}^{M} \mu_{t,m} g_{t,m}(\boldsymbol{x}_t) - \frac{\delta_t \|\boldsymbol{\mu}_t\|_2^2}{2}, \quad (23)$$

where $\boldsymbol{\mu}_t = (\mu_{t,m} : \forall m \in \mathcal{M})$ are the Lagrangian multipliers associated with the minimum guarantee constraints in slot t. Specifically, $\mu_{t,m}$ could be interpreted as the penalty of unit violation for the m-th minimum guarantee requirement in slot t. Moreover, $\|\cdot\|_2$ denotes the Euclidean norm, and δ_t is the regularization parameter (to be specified later).

Based on the Lagrangian function (23), we introduce the four major components of weighted alpha-fair online allocation framework $\mathcal{U}$.

1) Weighted Alpha-Fair Allocation: In each slot t, the weighted alpha-fair online allocation framework $\mathcal{U}$ will adopt the $(\boldsymbol{w}_t, \alpha_t)$-fair allocation $\hat{\boldsymbol{x}}_t$ defined in (20). Recall that $\hat{\boldsymbol{x}}_t$ has a closed-form expression in Proposition 1, thus this step exhibits a low computational complexity. The subsequent two components will specify the fairness levels $\{\alpha_t : \forall t \in \mathcal{T}\}$ and the weight vectors $\{\boldsymbol{w}_t : \forall t \in \mathcal{T}\}$, respectively,

2) Merit-Based Weight: The weighted alpha-fair online allocation framework $\mathcal{U}$ will adopt the merit-based weight based on the Lagrangian function (23). Intuitively, minimizing the Lagrangian function implicitly reduces the efficiency loss and the minimum guarantee violation. Therefore, at the end of each time slot, we update the weight vector according to the partial derivative of the Lagrangian function as follows

$$w_{t+1,n} = w_{t,n} \cdot \exp\left(-\frac{\partial \mathcal{L}_t(\hat{\boldsymbol{x}}_t; \boldsymbol{\mu}_t)}{\partial x_n}\right), \ \forall n \in \mathcal{N}, \quad (24)$$

where $\hat{\boldsymbol{x}}_t$ is the adopted allocation decision in slot t. The updating rule in (24) is inspired by Proposition 2

3) Increasing Fairness: The weighted alpha-fair online allocation framework $\mathcal{U}$ adopts the following fairness levels

$$\alpha_t \triangleq t^{\epsilon} G \sqrt{\frac{1+M}{2\ln(N)}}, \quad \forall t \in \mathcal{T}, \quad (25)$$

where $\epsilon \in (0, 1)$ is determined by the system operator. We have three remarks regrading the above expression (25).

- Note that α_t is increasing in t for any $\epsilon \in (0, 1)$. Moreover, $\epsilon \in (0, 1)$ is a tunable parameter in this framework. As we will see in Section V-C, it actually determines the three-way tradeoff among the fairness level, the efficiency loss (in the regret form), and the minimum guarantee violation.
- The expression (25) requires little prior knowledge. To calculate the corresponding fairness level in (25), the system operator only needs to possess the partial derivative bound G, the number of minimum guarantee requirements M, and the number of requesters N. Hence this step is simple for practical implementation.
- The expression in (25) is determined with the goal of achieving the sub-linear efficiency loss (in a regret form) and the sub-linear minimal guarantee violation. We refer interested readers to the detailed proof in appendix (see supplementary material).

Note that the achieved fairness may be different from what the practitioners expect to realize. To certain degree, one could take ϵ as what the practitioners expect to achieve. Furthermore, practitioners could also use fixed fairness level if T is known, which does not change the major analysis.

Algorithm 3 Weighted Alpha-Fair Online Allocation $\mathcal{U}$ With Long-Term Constraints

Input : A, N, M, and G
Output: Allocation decisions $\hat{\boldsymbol{x}}_t$ for each slot t.
Determine $\epsilon \in (0, 1)$
Initialize $\boldsymbol{w}_1 = \mathbf{1}_N$ and $\boldsymbol{\mu}_1 = \mathbf{0}_M$
for $t = 1$ **to** T **do**
  /* At the beginning of this slot */
  Calculate the fairness level α_t according to (25).
  Adopt the $(\boldsymbol{w}_t, \alpha_t)$-fair allocation $\hat{\boldsymbol{x}}_t$ in (20).
  /* At the end of this slot */
  Observe the revealed loss function $f_t(\cdot)$ and the minimum guarantee requirement $\{g_{t,m}(\cdot) : \forall m \in \mathcal{M}\}$
  Update the weight vector $\boldsymbol{w}_{t+1}$ according to (24).
  Update $\boldsymbol{\mu}_{t+1}$ according to (26).

4) Violation Penalty: To ensure the minimum guarantee requirement in the long-term, it is necessary to dynamically increase (or decrease, respectively) the penalty $\boldsymbol{\mu}_t$ if the current violation is large (or small, respectively). Hence the weighted alpha-fair online allocation framework $\mathcal{U}$ will update the Lagrangian multipliers $\boldsymbol{\mu}_{t+1}$ according to

$$\begin{aligned}\mu_{t+1,m} &= \left[\mu_{t,m} + \beta_t \cdot \frac{\partial \mathcal{L}_t(\hat{\boldsymbol{x}}_t; \boldsymbol{\mu}_t)}{\partial \mu_m}\right]^+, \\ &= \left[\mu_{t,m} + \beta_t \cdot \left(g_{t,m}(\hat{\boldsymbol{x}}_t) - \delta_t \mu_{t,m}\right)\right]^+, \end{aligned} \quad (26)$$

where the second equality follows directly from the definition in (23). Moreover, β_t is a step-size parameter, and δ_t is the regularization parameter in the Lagrangian function (23). The weighted alpha-fair online allocation framework $\mathcal{U}$ will determine the two parameters β_t and δ_t based on the current fairness level α_t according to

$$\delta_t = \frac{3(1+M)G^2 A}{\alpha_t}, \quad (27a)$$

$$\beta_t = \frac{1}{(t+1)\delta_t}. \tag{27b}$$

The above expressions are determined with the goal of reducing the cumulative regret $Reg_T(\mathcal{U})$ and the violation $Vio_T(\mathcal{U})$. We refer interested readers to the proof in appendix (see supplementary material).

So far, we have introduced the four major components. Algorithm 3 summarizes the weighted alpha-fair online allocation framework $\mathcal{U}$. Specifically, the input parameters include the resource amount A, the number of requesters N, the number of minimal guarantee requirements M, and the partial derivative bound G. The weighted alpha-fair online allocation framework $\mathcal{U}$ first specifies the parameter $\epsilon \in (0,1)$ in Line 3. We then initialize the weight vector $\boldsymbol{w}_1$ and the Lagrangian multipliers $\boldsymbol{\mu}_1$ as the all-one vector and all-zero vector, respectively. Each time slot consists of two phases:

- *Lines 4-5:* At the beginning of slot t, we calculate the fairness level α_t according to (25), and then adopt the $(\boldsymbol{w}_t, \alpha_t)$-fair allocation $\hat{\boldsymbol{x}}_t$ in (20).
- *Lines 6-8:* At the end of slot t, the loss function $f_t(\cdot)$ and the minimum guarantee requirements $\{g_{t,m}(\cdot) : \forall m \in \mathcal{M}\}$ are revealed. We then calculate the weight vector $\boldsymbol{w}_{t+1}$ and the Lagrangian multipliers $\boldsymbol{\mu}_{t+1}$ for the next slot according to (24) and (26), respectively.

Before presenting the theoretical performance of weighted alpha-fair online allocation framework, let us highlight its three advantages. First, Algorithm 3 is adaptive in time and does not require the prior knowledge on the total number of slots T. Hence it is simple for practical implementation, and the system operator does not need to perform the doubling-trick [27]. Second, Algorithm 3 requires very limited prior knowledge. Specifically, the input parameters only include the number of requesters N, the number of constraints M, the resource amount A, and the derivative bound G. Third, Algorithm 3 has a low computational complexity and does not involve high-dimensional projection operation. The above three aspects correspond to our first remark for the weighted alpha-fair online allocation framework $\mathcal{U}$.

Remark 1: The weighted alpha-fair online allocation framework in Algorithm 3 is an adaptive online allocation framework, which requires limited prior knowledge and has a low computation complexity.

C. Performance Analysis

Now we analyze the theoretical performance achieved by the weighted alpha-fair online allocation framework $\mathcal{U}$. As we will show in Theorem 3, our proposed weighted alpha-fair online allocation framework $\mathcal{U}$ achieves a sub-linear efficiency loss and a sub-linear minimum guarantee violation. The major analysis for this result consists of two steps. For notation simplicity, we define $J_T(\boldsymbol{\alpha})$, $J_T(\boldsymbol{\beta})$, and $J_T(\boldsymbol{\delta})$ as follows

$$J_T(\boldsymbol{\alpha}) \triangleq \sum_{t=1}^{T} \frac{1}{\alpha_t}, \quad J_T(\boldsymbol{\beta}) \triangleq \sum_{t=1}^{T} \beta_t, \quad J_T(\boldsymbol{\delta}) \triangleq \sum_{t=1}^{T} \delta_t. \tag{28}$$

Note that the three formulas in (28) may scale in T. Nevertheless, Lemma 1 indicates that all the three formulas have sub-linear upper bound. The proof follows directly from the inequality condition $\sum_{t=1}^{T} t^{-z} \leq \frac{T^{1-z}}{1-z}$ for any $z \in (0,1)$.

Lemma 1: Given the parameters in (25) and (27), we have

$$J_T(\boldsymbol{\alpha}) \leq \frac{1}{G}\sqrt{\frac{2\ln(N)}{1+M}} \cdot \frac{T^{1-\epsilon}}{1-\epsilon}, \tag{29a}$$

$$J_T(\boldsymbol{\beta}) \leq \frac{1}{3GA\sqrt{2(1+M)\ln(N)}} \cdot \frac{T^{\epsilon}}{\epsilon}, \tag{29b}$$

$$J_T(\boldsymbol{\delta}) \leq 3GA\sqrt{2(1+M)\ln(N)} \cdot \frac{T^{1-\epsilon}}{1-\epsilon}. \tag{29c}$$

Next we present the three-step performance analysis for the weighted alpha-fair online allocation framework $\mathcal{U}$.

Step 1: Based on the Lagrangian function defined in (23), we derive an upper bound for $\sum_{t=1}^{T}[\mathcal{L}_t(\hat{\boldsymbol{x}}_t;\boldsymbol{\mu}) - \mathcal{L}_t(\hat{\boldsymbol{x}}_t;\boldsymbol{\mu}_t)]$ and $\sum_{t=1}^{T}[\mathcal{L}_t(\hat{\boldsymbol{x}}_t;\boldsymbol{\mu}_t) - \mathcal{L}_t(\boldsymbol{x};\boldsymbol{\mu}_t)]$ in Lemma 2 and Lemma 3, respectively. The proof is given in appendix (see supplementary material).

Lemma 2: For any $\boldsymbol{\mu} \in \mathbb{R}_+^M$, we have

$$\sum_{t=1}^{T} \mathcal{L}_t(\hat{\boldsymbol{x}}_t;\boldsymbol{\mu}) - \mathcal{L}_t(\hat{\boldsymbol{x}}_t;\boldsymbol{\mu}_t) \leq \frac{\|\boldsymbol{\mu}\|_2^2}{2}\left(\frac{1}{\beta_1} - \delta_1\right) + MNA^2G^2J_T(\boldsymbol{\beta}) + \sum_{t=1}^{T}\beta_t\delta_t^2\|\boldsymbol{\mu}_t\|_2^2, \tag{30}$$

where $\{\hat{\boldsymbol{x}}_t : \forall t \in \mathcal{T}\}$ and $\{\boldsymbol{\mu}_t : \forall t \in \mathcal{T}\}$ are generated by our proposed weighted alpha-fair online allocation framework $\mathcal{U}$.

Lemma 3: For any feasible allocation $\boldsymbol{x} \in \mathcal{X}$, we have

$$\sum_{t=1}^{T} \mathcal{L}_t(\hat{\boldsymbol{x}}_t;\boldsymbol{\mu}_t) - \mathcal{L}_t(\boldsymbol{x};\boldsymbol{\mu}_t) \leq \left[\alpha_T\ln(N) + \frac{(1+M)G^2}{2}\left(J_T(\boldsymbol{\alpha}) + \sum_{t=1}^{T}\frac{\|\boldsymbol{\mu}_t\|_2^2}{\alpha_t}\right)\right]A, \tag{31}$$

where $\{\hat{\boldsymbol{x}}_t : \forall t \in \mathcal{T}\}$ and $\{\boldsymbol{\mu}_t : \forall t \in \mathcal{T}\}$ are generated by our proposed weighted alpha-fair online allocation framework $\mathcal{U}$.

Step 2: Based on Lemma 2 and Lemma 3, we show that the cumulative regret $Reg_T(\mathcal{U})$ and violation $Vio_T(\mathcal{U})$ satisfy the following inequality

$$Reg_T(\mathcal{U}) + \frac{\left[Vio_T(\mathcal{U})\right]^2}{2M\left[\frac{1}{\beta_1} - \delta_1 + J_T(\boldsymbol{\delta})\right]} \leq \left[\alpha_T\ln(N) + \frac{(1+M)G^2J_T(\boldsymbol{\alpha})}{2}\right]A + MNA^2G^2J_T(\boldsymbol{\beta}). \tag{32}$$

Step 3: Rearranging terms in (32) and substituting Lemma 1 lead to our main result in Theorem 3.

Theorem 3: For any $\epsilon \in (0,1)$, the weighted alpha-fair online allocation framework $\mathcal{U}$ in Algorithm 3 achieves the following efficiency loss in the regret form and minimum guarantee violation

$$Reg_T(\mathcal{U}) \leq AG \cdot \Omega_T(\epsilon),$$
$$Vio_T(\mathcal{U}) \leq AG\sqrt{\frac{24MCT^{2-\epsilon}}{1-\epsilon}\left[\sqrt{2N} + \frac{\Omega_T(\epsilon)}{T}\right]}, \tag{33}$$

where $\Omega_T(\epsilon) \sim O(T^{\max\{\epsilon,1-\epsilon\}})$ is given by

$$\Omega_T(\epsilon) \triangleq \left(T^{\epsilon} + \frac{T^{1-\epsilon}}{1-\epsilon}\right) \cdot C + \frac{T^{\epsilon}}{\epsilon} \cdot \frac{MN}{6C}, \tag{34}$$

and $C \triangleq \sqrt{(1+M)\ln(N)/2}$ is a constant.

Theorem 3 shows that the weighted alpha-fair online allocation framework $\mathcal{U}$ can simultaneously achieve the sub-linear efficiency loss in the regret form and the sub-linear minimum guarantee violation. That is, the adopted *weighted alpha-fair* allocations $\{\hat{\boldsymbol{x}}_t : \forall t \in \mathcal{T}\}$ in our framework perform asymptotically as well as the offline optimal/efficient benchmark $\boldsymbol{x}^{\star}$ in Definition 3. This is our second remark for the proposed framework.

Remark 2: The weighted alpha-fair online allocation framework in Algorithm 3 ensures the asymptotic efficiency and minimum guarantee by adopting the weighted alpha-fair allocations instantaneously in each time slot.

Theorem 3 also indicates that the tunable parameter $\epsilon \in (0,1)$ enables the system operator to flexibly control the three-way tradeoff among the fairness level, the system efficiency loss, and the minimum guarantee violation. The detailed elaborations are as follows:

- *Fairness Level:* Recall that the weighted alpha-fair online allocation framework $\mathcal{U}$ adopts the increasing fairness levels. Specifically, (25) shows that the adopted fairness level scales in the order $O(T^{\epsilon})$. A larger parameter $\epsilon \in (0,1)$ corresponds to a faster increasing speed for the fairness levels.
- *Efficiency Loss:* Theorem 3 indicates that the system efficiency loss (in the regret form) is upper bounded by $AG\Omega_T(\epsilon)$, where $\Omega_T(\epsilon)$ scales in T according to the sub-linear order $O\left(\frac{T^{1-\epsilon}}{1-\epsilon} + \frac{T^{\epsilon}}{\epsilon}\right)$. To reduce this sub-linear bound, the parameter $\epsilon \in (0,1)$ should not be excessively small or large. When ϵ is decreasing to 0, our framework $\mathcal{U}$ tends to adopt the less fair allocations, which lead to insufficient exploration (thus excessive exploitation). When ϵ is increasing to 1, our framework $\mathcal{U}$ actually enforces the more fair allocations, which result in excessive exploration (thus insufficient exploitation). Both the two cases above are undesirable.
- *Minimum Guarantee Violation:* Theorem 3 indicates that the violation upper bound scales in T according to the order $O\left(\sqrt{T^{2-\epsilon}/(1-\epsilon)}\right)$. Similarly, the parameter $\epsilon \in (0,1)$ should not be excessively small or large to reduce this sub-linear upper bound.

The above discussions lead to our third remark for the weighted alpha-fair online allocation framework $\mathcal{U}$.

Remark 3: The weighted alpha-fair online allocation framework has a tunable parameter $\epsilon \in (0,1)$, which controls the following three-way tradeoff among fairness level, efficiency loss, and minimum guarantee violation:

$$O\left(T^{\epsilon}\right),\ O\left(\frac{T^{1-\epsilon}}{1-\epsilon} + \frac{T^{\epsilon}}{\epsilon}\right),\ O\left(\sqrt{\frac{T^{2-\epsilon}}{1-\epsilon}}\right). \tag{35}$$

So far, we have presented the weighted alpha-fair online allocation framework as well as the theoretic performance. All the above results take into account the minimum guarantee requirement characterized by a total of M constraints. Next we demonstrate how to use it in a networking application.

VI. Case Study

We demonstrate our proposed framework via case study, focusing on caching capability allocation in Section VI-A and size-based scheduling in Section VI-B.

A. Caching Capacity Allocation

We demonstrate how to apply our proposed weighted alpha-fair online allocation framework $\mathcal{U}$ in caching capacity allocation problem.

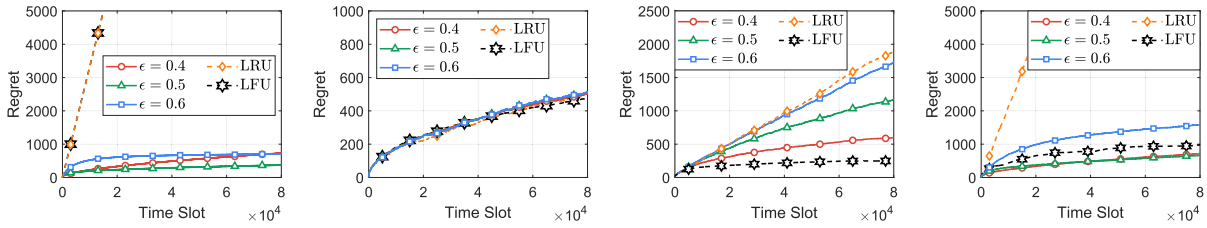
1) Background: File caching can significantly reduce the user request delay and the routing cost by hosting the popular files in a nearby cache [34]. The cache can typically store only a small portion of the file library due to its limited storage capacity [35]. A caching policy needs to determine which files should be stored without the prior knowledge on the requests or the file popularity. Hence, it is necessary to allocate the caching capacity in an online fashion [36]. Moreover, the efficiency of a caching policy is usually evaluated based on the cache-hit-ratio, which measures the fraction of requests met locally by the cache. There are some well-known caching policies, such as Least-Recently-Used (LRU) and Least-Frequently-Used (LFU). Their performance depends on the file request characteristics. For example, LFU usually performs well in stationary request. Nonetheless, the previous studies (e.g., Lemma 3.7 in [37]) have shown that LRU and LFU cannot achieve a sub-linear regret (i.e., the efficiency loss). Next we demonstrate how to apply our weighted alpha-fair online allocation framework to the file caching problem.

2) Formulation: We consider a file library with a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of N files, and let b_n denote the size (e.g., in MB) of file n. Moreover, we consider a cache with the storage capacity $A < \sum_{n=1}^{N} b_n$. The caching configuration will be iteratively adjusted during a long operation horizon. We partition the time horizon such that each slot corresponds to a single request [38]. We let $\boldsymbol{\sigma}_t = (\sigma_{t,n} \in \{0,1\} : \forall n \in \mathcal{N})$ denote the request in slot t, where $\sigma_{t,n} = 1$ indicates the request on file n in slot t. Note that we have $\sum_{n=1}^{N} \sigma_{t,n} = 1$. Furthermore, we let $\boldsymbol{x}_t \in \mathcal{X}$ denote the caching capacity allocation, thus $\min(x_{t,n}/b_n, 1)$ represents the fraction of file n cached in slot t. To improve the cache-hit-ratio, we consider the following loss function

$$f_t(\boldsymbol{x}_t) \triangleq -\sum_{n=1}^{N} \sigma_{t,n} v_n \min\left(\frac{x_{t,n}}{b_n}, 1\right), \tag{36}$$

and make two-fold elaborations on it:

- First, $v_n \geq 0$ represents the perceived valuation if the entire file n is hit in the cache (i.e., $x_{t,n} = b_n$). The vector $\boldsymbol{v} = (v_n : \forall n \in \mathcal{N})$ is determined by the cache operator and indicates the preference across the file library $\mathcal{N}$.
- Second, $\min(x_{t,n}/b_n, 1)$ implies that there is no additional benefit if the allocated caching capacity exceeds the file size. Note that this term is not differentiable at $x_{t,n} = b_n$, while zero is a sub-gradient at this point.



(a) Periodic request (b) Stationary request: uniform (c) Stationary request: Zipf (d) Mixed request

Fig. 1. File caching results (The solid curves represent weighted alpha-fair online allocation framework with $\epsilon \in \{0.4, 0.5, 0.6\}$).

We will use this sub-gradient in the merit-based weight updating (24).

Minimizing the loss function in (36) is equivalent to maximize the *cache-hit-ratio* when we take $\boldsymbol{v} = \mathbf{1}_N$. Later on, we will compare the cache-hit-ratio achieved by LFU, LRU, and our weighted alpha-fair online allocation framework $\mathcal{U}$.

3) Numerical Results: We compare the weighted alpha-fair online allocation framework to the typical caching policies LRU and LFU under different request characteristics. Specifically, we consider $N = 100$ files with the equal unit size and the same valuation $\boldsymbol{v} = \mathbf{1}_N$. Fig. 1 plots the average results of multiple simulation runs under the caching capacity $A = 10$. The four sub-figures correspond to different file request characteristics.

Periodic Request: Fig. 1(a) plots the results under the periodic file requests, i.e., $\{1, 2, \ldots, 30, 1, 2, \ldots, 30, \ldots\}$. Previous studies (e.g., Lemma 3.7 in [37]) have shown that LRU and LFU cannot achieve a sub-linear regret under this type of periodic request. As shown in Fig. 1(a), the two dash curves increase linearly. Furthermore, the other three solid curves in Fig. 1(a) correspond to our weighted alpha-fair online allocation framework with $\epsilon \in \{0.4, 0.5, 0.6\}$, respectively. Overall, the three curves increases in a sub-linear fashion. Note that the case $\epsilon = 0.5$ (i.e., the triangle curve) achieves a smaller regret than the other two cases in the long-term. This observation matches with the regret upper bound in Theorem 2, where $\epsilon = 0.5$ minimizes the order of the regret upper bound $O\left(\frac{T^{1-\epsilon}}{1-\epsilon} + \frac{T^{\epsilon}}{\epsilon}\right)$.

Stationary Request: Fig. 1(b) and Fig. 1(c) focus on the stationary file request pattern with the uniform popularity and Zipf popularity, respectively. In general, LFU performs optimally in response to the stationary requests, since it estimates the stationary file popularity based on the observed request frequency. As shown in Fig. 1(b) and Fig. 1(c), LFU achieves the smallest regret. In Fig. 1(b), the five curves almost overlap with each other. This means that both the weighted alpha-fair online allocation framework and the LRU caching policy can perform as well as LFU when the files are equally popular. In Fig. 1(c), LFU (i.e., the star curve) achieves a smaller regret than other cases under the Zipf popularity. Comparing the three solid curves shows that a smaller ϵ could reduce the regret. This observation is slightly different from the regret bound in Theorem 2. The reason is that Theorem 2 focuses on the non-stationary environment, and the corresponding regret bound may not be tight in the stationary case.

Mixed Request: Fig. 1(d) considers the mixed request, where we randomly insert some periodic request sequence into the stationary file request with Zipf popularity. Comparing the three solid curves to the star curve shows that the weighted alpha-fair online allocation framework outperforms LFU.

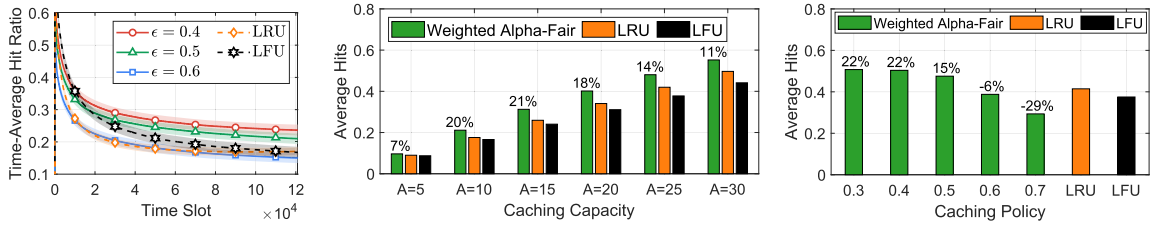
Shot Noise Request: The above file request patterns are kind of artificial. Now we evaluate the caching performance based on the shot noise request model [39], which captures the ephemeral YouTube video requests. Fig. 2 shows the results, where the vertical axis in the three sub-figures represents the time-average hits in the long-term.

- Fig. 2(a) plots the results under the caching capacity $A = 10$. In this case, the weighted alpha-fair online allocation framework with $\epsilon = 0.4$ (i.e., circle curve) and $\epsilon = 0.5$ (i.e., triangle curve) can outperform the classic caching policies LFU (i.e., star curve) and LRU (i.e., diamond curve) in the long-term.
- Fig. 2(b) plots the time-average hits achieved by the weighted alpha-fair online allocation framework with $\epsilon = 0.5$, LRU, and LFU under various caching capacities $A \in \{5, 10, 15, 20, 25, 30\}$. The percentage values at the top of green bars (i.e., weighted alpha-fair online) represent the improvement compared to the orange bars (i.e., LRU). Overall, the weighted alpha-fair online allocation framework improves the average hits up to 20% compared to the state-of-the-art caching policies.
- Fig. 2(c) plots the time-average hits achieved by the weighted alpha-fair online allocation framework with $\epsilon \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$, LRU, and LFU Note that when the increasing speed of fairness level is large (e.g., $\epsilon \in \{0, 6, 0.7\}$), the weighted alpha-fair online allocation framework may perform worse off than the state-of-the-art caching policies. Nevertheless, the weighted alpha-fair online allocation framework with a proper fairness criteria adoption (e.g., $\epsilon \in \{0.3, 0.4\}$) improves the average hits up to 23% compared to state-of-the-art caching policies.

B. Size-Based Scheduling

We demonstrate how to apply the weighted alpha-fair online allocation framework $\mathcal{U}$ in size-based flow scheduling.

1) Background: Flow scheduling is a critical issue in the modern data-center network. Most active flow scheduling schemes are partly implemented in the network switches. The bandwidth allocation scheme installed on the switches



(a) Case of $A = 10$ (b) Impact of caching capacity (c) Impact of $\epsilon \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$

Fig. 2. Results of shot noise request model.

plays a significant role on reducing the flow completion time (FCT). The size-based scheduling that prioritizes short flows is effective for reducing the average FCT. Typical solutions include pFabric [40], ADS [41], SLR [42], to name a few. However, the flow size information is often *prior unknown* for the network switches. That is, the size-based scheduling requires that the switch should allocate its bandwidth to multiple classes of flows in an online manner. Next we introduce the basic formulation of size-based scheduling according to the online allocation framework.

2) Formulation: We consider a data-center network with a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of N classes of flows. The N classes of flows are generated by different applications (e.g., user request and replication), thus may have different and possibly time-varying average flow sizes. We focus on a generic network switch with the bandwidth A. The switch will configure the bandwidth allocation to the N classes of flows at the beginning of each slot (e.g., every 10 seconds). We let $\boldsymbol{x}_t \in \mathcal{X}$ denote the bandwidth allocation in slot t. Next we introduce the efficiency measurement and the minimum guarantee requirement in size-based scheduling.[4]

Efficiency Measurement: The size-based scheduling aims to prioritize the short flows and reduce the average FCT. For this goal, we follow the previous work NUMFabric [43] and consider the following loss function in this use case:

$$f_t(\boldsymbol{x}_t) = -\sum_{n=1}^{N} \frac{x_{t,n}}{s_{t,n}}, \tag{37}$$

where $s_{t,n}$ denotes the average size of the scheduled class-n flows in slot t. The vector $\boldsymbol{s}_t = (s_{t,n} : \forall n \in \mathcal{N})$ represents the flow size information in slot t, which is disclosed at the end of this slot. Note that minimizing (37) turns out to allocate more bandwidth to the flow class of a smaller average flow size. This intuition coincides with the Shortest-Flow-First scheduling policy, thus the loss function in (37) is in favor of reducing the average FCT [43]. In this case, minimizing the loss function in (37) is equivalent to maximize the number of completed messages.

Minimum Guarantee Requirement: The loss function in (37) implicitly induces the bandwidth allocation outcome that prioritizes short flows. To prevent the starvation of long flows, we also need to take into account a total of N minimum guarantee requirements, one for each flows. Mathematically, we let ρ_n denote the least bandwidth allocated flow $n \in \mathcal{N}$, and consider N constraints $\{g_n(\boldsymbol{x}) \leq 0, \forall n \in \mathcal{N}\}$, where $g_n(\boldsymbol{x})$ is given by

$$g_n(\boldsymbol{x}) = \rho_n - x_n. \tag{38}$$

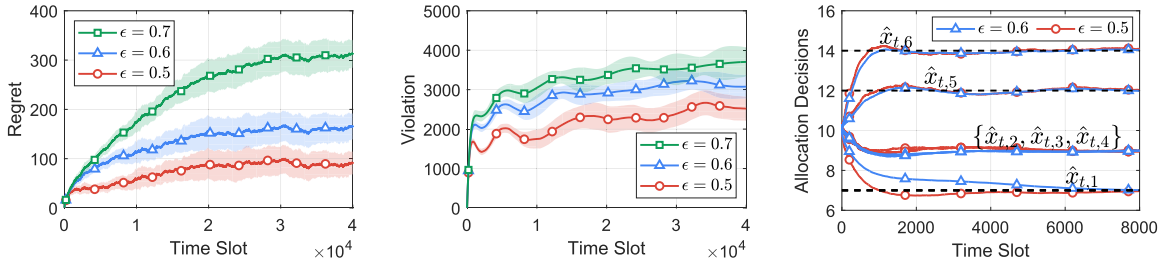
Note that if the size-based scheduling policy ensures the above minimum guarantee constraints in the long-term, then the time-average bandwidth allocated to the class-n flows is no less than ρ_n. Therefore, the above minimum guarantee requirements could implicitly avoid flow starvation in the size-based scheduling problem.

3) Simulation Results: We consider $N = 6$ classes of flows and randomly generate $\boldsymbol{s}_t$ according to the heavy-tail Pareto distribution with the mean values $\{60, 50, 50, 50, 50, 50\}$. That is, the class-1 flows are longer than other classes on average (which is prior unknown yet). Moreover, we suppose that the total bandwidth is $A = 60$ Gbps, and set the minimum guarantee as $\boldsymbol{\rho} = [7, 7, 7, 7, 12, 14]$. That is, the class-5 and class-6 flows are supposed to obtain a better performance guarantee than others in the long-term. We apply the weighted alpha-fair online allocation framework $\mathcal{U}$ to the size-based scheduling problem under different parameters $\epsilon = \{0.5, 0.6, 0.7\}$. Fig. 3 shows the average results of multiple simulation runs.

Fig. 3(a) and Fig. 3(b) plot the efficiency loss in the regret form and the minimum guarantee violation, respectively. The three curves in each sub-figure correspond to different parameters $\epsilon = \{0.5, 0.6, 0.7\}$. Recall that a larger ϵ means a greater fairness increasing speed. As shown in Fig. 3(a) and Fig. 3(b), a greater fairness increasing speed results in larger efficiency loss and more violation. Furthermore, the fluctuation of curves in Fig. 3(b) is due to the back-and-forth movement of the allocation decisions around the minimum guarantee thresholds $\boldsymbol{\rho}$ (as shown in Fig. 3(c)).

Fig. 3(c) plots the allocation decisions $\{\hat{\boldsymbol{x}}_t : \forall t \in \mathcal{T}\}$ for cases $\epsilon \in \{0.5, 0.6\}$. In the first time slot, the N classes of flows equally share the bandwidth for both cases. During the remaining slots, the two cases are slightly different: The class-1 flows are longer than others on average, thus $\hat{x}_{t,1}$ is gradually decreasing and converging to $\rho_1 = 7$ as shown in Fig. 3(c). The decreasing speed in case $\epsilon = 0.5$ is faster than that in case $\epsilon = 0.6$. This is because that the case $\epsilon = 0.6$ adopts more fair allocations, thus prioritizes short flows in a weaker strength. The class-5 and class-6 flows have stronger minimum guarantee requirements than others. Hence $\hat{x}_{t,5}$ and $\hat{x}_{t,6}$ gradually increase in t, and eventually converge to $\rho_5 = 12$ and $\rho_6 = 14$, respectively. Case $\epsilon = 0.5$

[4]Most commodity switches provide multiple FIFO queues [40]. The switch can use FIFO queues to implement desired bandwidth allocation.



(a) Efficiency loss in the regret form (b) Minimum guarantee violation (c) Allocation decisions $\{\hat{\boldsymbol{x}}_t : \forall t\}$

Fig. 3. Numerical results in size-based scheduling.

responses to the minimum guarantee requirement faster than the case $\epsilon = 0.6$. To sum up, our proposed weighted alpha-fair online allocation framework enables the network switches to prioritize short flows and avoid the flow starvation without requiring any prior information on the flow size distribution.

VII. Conclusion and Future Work

In this paper, we investigate an online learning meta-algorithm (i.e., MWU) from the perspective of weighted alpha-fair resource allocation. We unveil that MWU inadvertently adopts a weighted alpha-fair probability allocation in each slot. Specifically, the fairness level equals to the inverse of the step-size, while the weight of each action exhibits the exponential decay with respect to its incurred cumulative cost. The discovery indicates that appropriate weighted-alpha fair allocation (for the action-taking probability) can perform as well as the offline optimal outcome in online learning. Motivated by this, we further investigate whether one could solve more general online allocation problems (not limited to the one addressed by MWU) by properly customizing the weighted-alpha fair allocation in each slot. Hence we propose a weighted alpha-fair online allocation framework, which can ensure sub-linear regret through per-slot weighted alpha-fair allocations. We also demonstrate how to use this framework in caching capability allocation and size-based flow scheduling.

The key contribution of this work is the recognition of the new interpretation of MWU's behavior as implicitly following weighted alpha-fair allocation. This interpretation may not immediately lead to significant performance improvement (for regret), but motivates us to further investigate why only weighted alpha-fairness works. We also try other fairness concepts in online allocation, but only find weighted alpha-fairness can ensure a sub-linear regret. In the future, we will investigate whether/why weighted alpha-fairness is unique.

Our study in this paper focuses on the online allocation of divisible resources. It would be interesting to generalize the proposed framework to the case of indivisible resource. The indivisible resource may represent single wireless channel to be shared by multiple IoT sensors, or a single display position to be shared by multiple advertisers. In this case, multiple requesters have to share the single indivisible resource repetitively across different time slots. The major challenge in the online indivisible resource allocation is the *bandit feedback*. To address this challenge, we need to modify the problem formulation and the current weighted alpha-fair online allocation framework.

References

[1] Z. Wang, J. Ye, D. Lin, and J. C. S. Lui, "Achieving efficiency via fairness in online resource allocation," in *Proc. 23rd Int. Symp. Theory, Algorithmic Found., Protocol Design Mobile Netw. Mobile Comput.*, Oct. 2022, pp. 101–110.

[2] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: A meta-algorithm and applications," *Theory Comput.*, vol. 8, no. 1, pp. 121–164, 2012.

[3] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 556–567, 2000.

[4] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Mach. Learn.*, vol. 2, no. 4, pp. 285–318, Apr. 1988.

[5] N. Littlestone, *Mistake Bounds and Logarithmic Linear-Threshold Learning Algorithms*. Santa Cruz, CA, USA: University of California, 1989.

[6] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Inf. Comput.*, vol. 108, no. 2, pp. 212–261, Feb. 1994.

[7] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[8] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Apr. 1998.

[9] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[10] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proc. IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.

[11] M. Dehghan, L. Massoulié, D. Towsley, D. S. Menasché, and Y. C. Tay, "A utility optimization approach to network cache design," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1013–1027, Jun. 2019.

[12] E. Hargreaves, C. Agosti, D. Menasché, G. Neglia, A. Reiffers-Masson, and E. Altman, "Fairness in online social network timelines: Measurements, models and mechanism design," *Perform. Eval.*, vol. 129, pp. 15–39, Feb. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166531618302724

[13] J. F. Nash, "The bargaining problem," *Econometrica*, vol. 18, no. 2, pp. 155–162, Apr. 1950.

[14] E. Kalai and M. Smorodinsky, "Other solutions to Nash's bargaining problem," *Econometrica*, vol. 43, no. 3, pp. 513–518, May 1975.

[15] T. Lan, D. Kao, M. Chiang, and A. Sabharwal, "An axiomatic theory of fairness in network resource allocation," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[16] G. S. Paschos, A. Destounis, and G. Iosifidis, "Online convex optimization for caching networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 625–638, Apr. 2020.

[17] H. Luo, H. Li, S. Zhang, Y. Mao, and Z. Wang, "Achieving packet traceback by inferring AS-level topology based on cryptographic path identifiers," *IEEE Trans. Inf. Forensics Security*, early access, 2025, doi: 10.1109/TIFS.2025.3622069.

[18] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. Cognit. Commun. Netw.*, vol. 4, no. 2, pp. 257–265, Jun. 2018.
[19] J. Feldman, M. Henzinger, N. Korula, V. Mirrokni, and C. Stein, "Online stochastic packing applied to display ad allocation," in *Proc. Eur. Symp. Algorithms*. London, U.K.: Springer, 2010, pp. 182–194.
[20] T. Chen, Q. Ling, and G. B. Giannakis, "An online convex optimization approach to proactive network resource allocation," *IEEE Trans. Signal Process.*, vol. 65, no. 24, pp. 6350–6364, Dec. 2017.
[21] F. Li, J. Liu, and B. Ji, "Combinatorial sleeping bandits with fairness constraints," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 3, pp. 1799–1813, Jul. 2020.
[22] M. Mahdavi, T. Yang, and R. Jin, "Online decision making under stochastic constraints," in *Proc. NIPS workshop Discrete Optim. Mach. Learn.*, 2012, pp. 1–8.
[23] H. Yu, M. J. Neely, and X. Wei, "Online convex optimization with stochastic constraints," in *Proc. NeurIPS*, 2017, pp. 1427 –1437.
[24] K. Cai, X. Liu, Y.-Z.-J. Chen, and J. C. S. Lui, "Learning with guarantee via constrained multi-armed bandit: Theory and network applications," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5346–5358, Sep. 2023.
[25] R. Jenatton, J. Huang, and C. Archambeau, "Adaptive algorithms for online convex optimization with long-term constraints," in *Proc. ICML*, 2016, pp. 402–411.
[26] Z. Wang, Q. Meng, S. Zhang, and H. Luo, "Performance evaluation for latency-stability tradeoff in satellite-terrestrial integrated networks," *IEEE Trans. Netw.*, early access, 2025, doi: 10.1109/TON.2025.3578240.
[27] S. Bubeck, "Convex optimization: Algorithms and complexity," *Found. Trends Mach. Learn.*, vol. 8, nos. 3–4, pp. 231–357, 2015.
[28] E. Hazan, "Introduction to online convex optimization," *Found. Trends Optim.*, vol. 2, nos. 3–4, pp. 157–325, 2016.
[29] H. Zhang, Z. Wang, W. Lu, S. Zhang, and H. Luo, "Source routing for LEO mega-constellations based on Bloom filter," *IEEE Trans. Mobile Comput.*, vol. 24, no. 11, pp. 12487–12504, Nov. 2025.
[30] H. Zhang, Z. Wang, S. Zhang, Q. Meng, and H. Luo, "Link-identified routing architecture in space," *IEEE Trans. Netw. Sci. Eng.*, vol. 12, no. 1, pp. 392–408, Jan. 2025.
[31] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multiresource allocation: Fairness–efficiency tradeoffs in a unifying framework," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1785–1798, Dec. 2013.
[32] Z. Wang, X. Lai, S. Zhang, Q. Meng, and H. Luo, "Enabling Byzantine fault tolerance in access authentication for mega-constellations," *IEEE/ACM Trans. Netw.*, vol. 32, no. 6, pp. 5341–5355, Dec. 2024.
[33] S. Mannor, J. N. Tsitsiklis, and J. Y. Yu, "Online learning with sample path constraints," *J. Mach. Learn. Res.*, vol. 10, no. 20, pp. 569–590, 2009.
[34] Z. Wang, L. Gao, T. Wang, and J. Luo, "Monetizing edge service in mobile internet ecosystem," *IEEE Trans. Mobile Comput.*, vol. 21, no. 5, pp. 1751–1765, May 2022.
[35] Z. Wang, J. Ye, and J. C. S. Lui, "Toward large-scale hybrid edge server provision: An online mean field learning approach," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 8, pp. 2347–2360, Aug. 2022.
[36] Z. Wang, L. Gao, and J. Huang, "Socially-optimal mechanism design for incentivized online learning," in *Proc. IEEE INFOCOM - IEEE Conf. Comput. Commun.*, May 2022, pp. 1828–1837.
[37] G. Paschos, G. Iosifidis, and G. Caire, "Cache optimization models and algorithms," *Found. Trends® Commun. Inf. Theory*, vol. 16, no. 3–4, pp. 156–343, 2020.
[38] Z. Wang, Q. Meng, S. Zhang, and H. Luo, "Incentivizing fresh information acquisition via age-based reward," *IEEE Trans. Netw.*, vol. 33, no. 3, pp. 1174–1188, Jun. 2025.
[39] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal locality in today's content caching: Why it matters and how to model it," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 5, pp. 5–12, Nov. 2013.
[40] M. Alizadeh et al., "pfabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 435–446, 2013.
[41] Z. Wang, J. Ye, D. Lin, Y. Chen, and J. C. S. Lui, "Approximate and deployable shortest remaining processing time scheduler," *IEEE/ACM Trans. Netw.*, vol. 30, no. 3, pp. 1368–1381, Jun. 2022.
[42] Q. Meng et al., "Switch-assistant loss recovery for RDMA transport control," *IEEE/ACM Trans. Netw.*, vol. 32, no. 3, pp. 2069–2084, Jun. 2024.
[43] K. Nagaraj, S. Chinchali, D. Bharadia, H. Mao, M. A. Attar, and S. Katti, "NUMFabric: Fast and flexible bandwidth allocation in datacenters," *ACM SIGCOMM*, pp. 188 –201, 2016.

Zhiyuan Wang received the B.Eng. degree in information engineering from Southeast University, Nanjing, in 2016, and the Ph.D. degree from the Department of Information Engineering, The Chinese University of Hong Kong, in 2019. From 2019 to 2021, he was a Post-Doctoral Fellow with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. He is currently an Associate Professor with the School of Cyber Science and Technology, Beihang University. His research interests include network architecture, routing protocol, algorithmic game theory, and online learning theory.

Jiancheng Ye (Senior Member, IEEE) received the B.E. degree in network engineering from Sun Yat-sen University, Guangzhou, China, in 2008, the M.Phil. degree in computer science and engineering from The Hong Kong University of Science and Technology, Hong Kong, in 2011, and the Ph.D. degree in computer networking from The University of Hong Kong, Hong Kong, in 2018. He was a Software Engineer with Harmonic Inc. from 2011 to 2014, a Post-Doctoral Fellow with The University of Hong Kong from 2018 to 2019, and a Senior Researcher with Huawei Hong Kong Research Center from 2019 to 2024. He is currently an Associate Professor with the School of Computer Science and Engineering, Macau University of Science and Technology. His research interests include network congestion control, protocol design and optimization, edge computing, and machine learning for networks.

John C. S. Lui (Fellow, IEEE) received the Ph.D. degree in computer science from UCLA. After his graduation, he joined the IBM Laboratory and participated in research and development projects on file systems and parallel I/O architectures. He later joined the CSE Department, The Chinese University of Hong Kong (CUHK). He is currently the Choh-Ming Li Chair Professor with the Department of Computer Science and Engineering (CSE), CUHK. His current research interests include quantum internet and distributed quantum computing, online learning algorithms and applications, machine learning on network sciences and networking systems, large-scale storage systems, and performance evaluation theory. He is an Elected Member of the IFIP WG 7.3, a Fellow of ACM, a Senior Research Fellow of the Croucher Foundation, and a Fellow of Hong Kong Academy of Engineering (HKAES).