Achieving Efficiency via Fairness in Online Resource Allocation

Zhiyuan Wang
Beihang University
zhiyuanwang@buaa.edu.cn

Jiancheng Ye
Huawei Technologies Co., Ltd.
yejiancheng@huawei.com

Dong Lin
Huawei Technologies Co., Ltd.
lindong10@huawei.com

John C.S. Lui
The Chinese University of Hong Kong
cslui@cse.cuhk.edu.hk

ABSTRACT

The classic utility maximization framework studies the fairness-efficiency tradeoff in various resource allocation problems (e.g., bandwidth allocation). The weighted alpha-fair utility is a common utilitarian metric. However, this classic framework cannot tackle those allocation problems with the online decision-making requirement (e.g., caching capacity allocation under unknown requests). Existing studies on these online allocation problems largely follow the online learning approaches, thus inevitably overlook the allocation fairness. In this paper, we propose a novel utility maximization framework accommodating the online setting. The major challenge of designing this framework lies in the tight coupling between the desirable fairness guarantee and the unknown allocation efficiency. To tackle this, we integrate the weighted alpha-fair utility with the learning rationale, by properly devising the merit-based weights and the increasing fairness levels. Under our proposed framework, the utility-maximizing allocation in each time slot is weighted alpha-fair. Our framework also performs asymptotically as well as the offline optimal/efficient outcome. We demonstrate how this framework functions in two networking applications. In size-based scheduling, it enables 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.

CCS CONCEPTS

• **Theory of computation** → **Online learning theory**.

KEYWORDS

Online learning, online resource allocation, alpha-fair utility

ACM Reference Format:
Zhiyuan Wang, Jiancheng Ye, Dong Lin, and John C.S. Lui. 2022. Achieving Efficiency via Fairness in Online Resource Allocation. In *The Twenty-third International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc'22), October 17–20, 2022, Seoul, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3492866.3549724

Jiancheng Ye is the corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MobiHoc'22, October 17–20, 2022, Seoul, Republic of Korea
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9165-8/22/10…$15.00
https://doi.org/10.1145/3492866.3549724

1 INTRODUCTION

1.1 Background and Motivation

1.1.1 Fairness & Efficiency. Resource allocation is one of the classic research problems in computer and communication networks. It has been heavily studied in network bandwidth allocation under the *utility maximization framework* (e.g., [1–6]). Some studies on caching capacity allocation (e.g., [7]) and ad space allocation (e.g., [8]) also take the utility maximization into consideration. The fairness-efficiency tradeoff is usually the epitome of these studies.

- Fairness is an abstract metric for equity and has a wide range of definitions. Mo and Walrand in [4] introduce the *weighted alpha-fair utility* that generalizes several fairness definitions such as the proportional fairness (introduced by Nash in [9]) and the max-min fairness (e.g., [10]). Recently, Lan *et al.* in [5] show what it actually means for a larger alpha value to be a more fair allocation outcome.
- Efficiency is a rather concrete criteria, which is usually defined by the system operator to characterize the system performance of its interest. Hence the efficiency of different networking systems often has different mathematic forms. Some may even involve time-varying unknown parameters.

In general, maximizing the system efficiency and ensuring the allocation fairness are often inconsistent. Hence most studies (e.g., [1]-[8]) investigate how to balance the system efficiency and the allocation fairness in the system design.

1.1.2 Online Allocation. In many real-world applications, the resource allocation problems often involve unknown parameters due to the changing environment. The operator cannot accurately anticipate the system performance (i.e., the efficiency metric) until the unknown parameters are revealed. This means that the allocation problems exhibit the need of performing online estimation and optimization. We will refer to these problems as the "*online resource allocation*" (e.g., [11–19]), and briefly introduce several important applications within this framework:

- the caching capacity allocation for files with unknown request or file popularity (e.g., [11, 12]),
- the wireless channel allocation for multiple IoT sensors with unknown sensing data amount (e.g., [13]),
- the display space allocation for multiple advertisers with unknown click rates (e.g. [14]),

- and the job dispatching to multiple geo-distributed datacenters with unknown energy expenditure (e.g., [15]).

The above online allocation problems study how the operator allocates the resource to multiple competing requesters. Due to the environment uncertainty, the operator has to dynamically optimize its allocation decisions without the prior knowledge. Most studies above follow the online learning/optimization framework (e.g., [20–22]). This framework focuses on a metric called *regret*, which is defined as the system performance gap between the offline optimal/efficient allocation and the adopted online allocation decisions. That is, regret measures the *efficiency loss* in an online allocation problem, and a sub-linear regret implies that the adopted online allocation decisions are asymptotically efficient. For example, [11] uses online gradient decent in caching capacity allocation to achieve a sub-linear regret.

1.1.3 Fairness in Online Allocation. Despite the achievable asymptotic efficiency, simply following the online learning/optimization approaches may cause an *unfair* allocation outcome, which is unacceptable in many real-life applications. This drawback motivates us to reconsider the classic utility-guided framework (e.g., weighted alpha-fair utility), leading to the first fundamental question:

QUESTION 1. *Can one develop a novel utility maximization framework (with fairness guarantee) for the online setting?*

Ideally, the above framework should achieve asymptotic efficiency by adopting the utility-maximizing allocation (with desired fairness guarantee) in each time slot. In this case, *fairness is no longer taken as a fixed input parameter, but a variable used to ensure asymptotic efficiency.*[1] Such a framework is desirable but challenging to design for the online setting with unknown parameters. It requires us to integrate the utility-guided fairness adoption with the unknown parameter learning in a proper manner. The major challenge is to figure out the underlying connection between the fairness concept and the learning rationale. If the answer to Question 1 is yes, then it is natural for us to further investigate the second key question:

QUESTION 2. *What is the relationship between efficiency loss (in the regret form) and fairness level in online allocation?*

To address the two key questions, we will extend and generalize the classic weighted alpha-fair utility to the online setting by devising the merit-based weights and the increasing fairness levels. The two aspects implicitly integrate the online learning rationale (i.e., the exploration-exploitation tradeoff) into the weighted alpha-fairness adoption.

1.2 Main Results and Key Contributions

This paper studies the network resource allocation in the online setting with unknown environment. At the beginning of each time slot, the network operator allocates the resource to multiple competing requesters. At the end of this time slot, the uncertain environment is revealed, and the operator observes the allocation efficiency (i.e., objective) and the minimum guarantee requirement (i.e., constraints). The performance criteria of this problem involve three aspects: (1) allocation fairness, (2) efficiency loss in the regret form, and (3) minimum guarantee violation. Our goal is to develop a novel utility maximization framework (with the fairness guarantee) for the general online allocation problem.

The main results and key contributions are as follows:

- *A Novel Utility Maximization Framework:* We propose a utility-guided allocation framework for the online allocation problem. This framework is a non-trivial generalization of the classic weighted alpha-fair utility to the online setting. It properly integrates the weighted alpha-fairness with the online learning rationale. Under this framework, the utility-maximizing allocation in each time slot satisfies the weighted alpha-fairness, and can also achieve the desired long-term performance. To our knowledge, this is the first online allocation framework ensuring weighted alpha-fairness.
- *Connection between Weighted Alpha-Fairness and Learning Rationale:* We unveil the intrinsic connection between the classic weighted alpha-fairness and the universal learning rationale. Specifically, the weight vector corresponds to the exploitation based on previous feedbacks, while the fairness level (i.e., alpha) controls the magnitude of exploration over the non-stationary environment. This intuition guides us to devise the merit-based weights and the increasing fairness levels, which manipulate the exploitation and exploration in the online setting, respectively.
- *Theoretical Performance:* Our proposed framework is adaptive in time and has a low computation complexity. It achieves a sub-linear efficiency loss in the regret form and a sub-linear minimum guarantee violation. That is, the weighted alpha-fair allocations in our framework perform asymptotically as well as the offline optimal/efficient outcome. Moreover, our framework has a tunable parameter $\epsilon \in (0,1)$, which controls the *three-way tradeoff* among fairness level with the order $O(T^{\epsilon})$, the efficiency loss with the order $O\Big(\frac{T^{1-\epsilon}}{1-\epsilon}+\frac{T^{\epsilon}}{\epsilon}\Big)$, and the minimum guarantee violation with the order $O\Big(\sqrt{T^{2-\epsilon}/(1-\epsilon)}\Big)$.
- *Experimental Results:* We demonstrate how the proposed framework functions in two networking applications. In size-based scheduling, our framework enables the network switches to prioritize short flows and avoid the 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 the cache-hit-ratio under the typical file request model.

The rest of this paper is as follows. Section 2 reviews related literatures. Section 3 introduces the system model. Section 4 presents our proposed framework. Section 5 provides the numerical results. We conclude this paper in Section 6.

2 LITERATURE REVIEW

There are two streams of studies related to this paper, which are summarized in Table 1.

[1]Similar idea also appears in the studies on recommender systems, where fairness is framed as "serendipity" in these literatures (e.g., [23][24]). That is, pro-actively controlling the fairness level actually determines how much serendipity we want.

Table 1: Literatures on offline/online allocation

Literatures		Fairness	Efficiency (loss)	Mini. Guaran.
[1]-[8]	Offline	*Fairness-efficiency Tradeoff*		NA
[11]-[15]	Online	NA	*Sub-linear*	NA
[16]-[19]		NA	*Sub-linear*	*Sub-linear*
This Paper		*Achieve sub-linear performance by adjusting fairness*		

2.1 Utility-Guided Allocation

Previous studies on utility-guided allocation focus on the fairness-efficiency tradeoff of various offline allocation problems. Typical applications include medical management (e.g., [25]), supply chain operation (e.g., [26]), air traffic control (e.g., [27]), and network bandwidth allocation (e.g., [1–6]). The *weighted alpha-fair utility*, introduced by Mo and Walrand in [4], is a prevalent utilitarian measurement and generalizes several fairness concepts. More recently, Lan *et al.* in [5] extend this scheme and demonstrate what it actually means for a larger alpha value to be more fair. Wong *et al.* in [6] extend this scheme to the multi-resource scenario. Furthermore, some studies on caching capacity allocation (e.g., [7]) and ad space allocation (e.g., [8]) 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. Hence the underlying allocation problem is still an offline one. Different from the studies above, this paper aims to generalize the weighted alpha-fair utility to the online setting, which is a non-trivial extension for the utility-guided allocation framework.

2.2 Learning-based Online Allocation

Most studies on online allocation problems follow the online learning/optimization approaches (e.g., [20–22]). Typical applications include file caching (e.g., [11][12]), channel access (e.g., [13]), ad display space allocation (e.g., [14]), and job dispatching (e.g., [15]). These studies focus primarily on improving the system performance (i.e., efficiency) and overlook the fairness issue. More recently, Li *et al.* in [16] propose a combinatorial bandit framework with fairness constraints. Nevertheless, the fairness constraints essentially represent the minimum guarantee requirement, which is also captured in [17–19] and our study in this paper. To our knowledge, there is no existing learning-based online allocation approach incorporating the weighted alpha-fairness concept [4].

2.3 Proposed Framework in This Paper

In this paper, we develop the first utility-guided allocation framework under the online setting. Our framework extends the weighted alpha-fair utility from the perspective of the merit-based weights and the increasing fairness levels. The two aspects implicitly integrate the universal learning rationale (i.e., exploitation and exploration) into the classic weighted alpha-fairness concept. In this way, 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.

3 SYSTEM MODEL

This paper aims to generalize the classic utility maximization framework to accommodate online allocation problems. Specifically, we will focus on the weighted alpha-fair utility introduced in [4]. We first review the online allocation problem in Section 3.1. We then introduce the utility-guided allocation framework and our problem formulation in Section 3.2 and Section 3.3, respectively.

3.1 Online Allocation Problem

Consider an online network resource/workload allocation problem, which has an operation horizon with a set $\mathcal{T} = \{1, 2, ..., T\}$ of T slots. The network operator will allocate a total of $A \in \mathbb{R}_+$ resource/workload to a set $\mathcal{N} = \{1, 2, ..., N\}$ of N nodes. Due to the environment uncertainty, the operator cannot anticipate the allocation 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 operator allocates a total of A resource/workload to the N nodes.
- At the end of time slot t, the uncertain environment is revealed, thus the operator perceives the feedbacks regarding the allocation performance.

Next, we characterize the allocation decision and the allocation performance for the general online allocation problem.

3.1.1 *Allocation Decision.*

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

$$\boldsymbol{x}_t \triangleq (x_{t,n} : \forall n \in \mathcal{N}) \,. \tag{1}$$

Given the total resource/workload amount A, the allocation decision $\boldsymbol{x}_t$ is chosen from the set $\mathcal{X}$, i.e.,

$$\mathcal{X} \triangleq \left(\boldsymbol{x} \in \mathbb{R}_+^N : \sum_{n=1}^{N} x_n = A \right). \tag{2}$$

Note that the equality condition in (2) does not reduce the generality in this problem. Essentially, it can explicitly capture both network workload allocation and network resource allocation. For workload allocation (e.g., job dispatching), the front-end job router will dispatch the jobs to the N server clusters [15]. For resource allocation (e.g., caching capacity allocation), a larger x_n means more allocated resource, thus will not make node n worse off [7].

3.1.2 *Allocation Performance.*

We model the system performance based on the allocation efficiency and the minimum guarantee. Both of them are possibly time-varying in the online setting.

Allocation Efficiency: We model the allocation efficiency based on the general loss functions. Such a loss function may represent the negative cache-hit-ratio or the energy expenditure. Mathematically, we let $f_t(\cdot)$ denote the loss function in slot t, and make two-fold elaborations. First, the operator does not know the value of $f_t(\cdot)$ until the end of slot t due to the uncertain environment (e.g., unknown file requests or electricity prices). Hence the operator has to determine $\boldsymbol{x}_t$ based only on the estimated information. Second, the loss functions $\{f_t(\cdot) : \forall t \in \mathcal{T}\}$ may vary over time possibly in a non-i.i.d. manner. In this work, we assume that the loss functions are convex but time-varying.

Minimum Guarantee: Besides the allocation efficiency, the operator needs to ensure some other critical minimum guarantee requirement for practical applications. Such a requirement may represent the time-average link utilization constraints or the queue 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 for real-world applications. To avoid the fundamental impossibility,[2] we follow the previous studies (e.g., [17–19]) and assume these constraints vary over time in an i.i.d. manner (i.e., stochastic constraints). Specifically, we consider a set $\mathcal{M} = \{1, 2, ..., M\}$ of M stochastic constraints

$$\{g_{t,m}(\boldsymbol{x}) \le 0 : \forall m \in \mathcal{M}\}, \tag{3}$$

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 i.i.d. random variable θ_t. Note that the random variables $\{\theta_t : \forall t \in \mathcal{T}\}$ capture the environment uncertainty. In practice, the operator does not possess the probability distribution and cannot predict the realizations.

3.1.3 Offline Benchmark. Based on the above discussions, we define the offline optimal/efficient allocation $\boldsymbol{x}^\star$ as follows.

Definition 1. *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 $\boldsymbol{x}^\star$, i.e.,*

$$\begin{aligned} \boldsymbol{x}^\star \triangleq \arg\min_{\boldsymbol{x} \in \mathcal{X}} \quad & \sum_{t=1}^{T} f_t(\boldsymbol{x}) \\ s.t. \quad & \mathbb{E}\left[\tilde{g}_m(\boldsymbol{x}; \theta_t)\right] \le 0, \quad \forall m \in \mathcal{M}. \end{aligned} \tag{4}$$

The total loss incurred by $\boldsymbol{x}^\star$ is the benchmark that we want to achieve in the online allocation. Next, we first elaborate why the classic utility maximization framework cannot attain this benchmark in Section 3.2. We then present the problem formulation of developing a novel utility-guided framework in Section 3.3.

3.2 Weighted Alpha-Fair Utility Maximization

The classic utility maximization framework tends to make the allocation decision in each slot by maximizing the total utility of all the nodes. Specifically, the utility function is usually *"predefined"* with different fairness criteria. There are different schemes to evaluate the allocation fairness. The *weighted alpha-fairness*, introduced by Mo and Walrand in [4], corresponds to a utility function that is parameterized by a weight vector $\boldsymbol{w} = (w_n \ge 0 : \forall n \in \mathcal{N})$ and a quantitative fairness level $\alpha \ge 0$. Given a feasible allocation decision $\boldsymbol{x} \in \mathcal{X}$, the corresponding weighted alpha-fair utility is

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

[2]Mannor *et al.* in [28] 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., [17–19]) focus on the case with the non-i.i.d. loss functions and the i.i.d. constraints.

The feasible allocation that maximizes the weighted alpha-fair utility $U(\boldsymbol{x}; \boldsymbol{w}, \alpha)$ satisfies the corresponding $(\boldsymbol{w}, \alpha)$-fairness [4], which is defined in Definition 2.

Definition 2. *Given the weight $\boldsymbol{w} \in \mathbb{R}_+^N$ and $\alpha \ge 0$, the following allocation outcome $\hat{\boldsymbol{x}}$ is $(\boldsymbol{w}, \alpha)$-fair.*

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

Based on Definition 2, let us elaborate the generality of the above fairness definition from two aspects:

- It provides a tunable fairness level $\alpha \ge 0$, and generalizes the proportional fairness when $\alpha = 1$ and the max-min fairness when $\alpha \to \infty$. Moreover, it is usually believed that a larger α value corresponds to a more fair allocation outcome [5].
- To a certain extent, the weighted alpha-fair utility can also capture the allocation efficiency in some offline allocation problems. For example, in network bandwidth allocation, $U(\boldsymbol{x}; \boldsymbol{w}, \alpha)$ degenerates into the (wighted) network throughput $\sum_{n \in \mathcal{N}} w_n x_n$ when we take $\alpha = 0$.

However, the second aspect above fails in the online allocation problems. This is because the allocation efficiency and the minimum guarantee introduced in Section 3.1 have more complicated structures (not always the linear sum), more importantly, it involves unknown information. Therefore, the classic weighted alpha-fair utility (5) is not able to reflect the online allocation performance. This requires us to generalize the weighted alpha-fair utility to online setting. Next we introduce the problem formulation.

3.3 Problem Formulation

We aim to design a novel utility maximization framework $\mathcal{U}$ for the general online 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 need to customize the weight vectors $\{\boldsymbol{w}_t : \forall t \in \mathcal{T}\}$ and the fairness levels $\{\alpha_t : \forall t \in \mathcal{T}\}$ of the weighted alpha-fair utility. That is, the sequence $\{(\boldsymbol{w}_t, \alpha_t) : \forall t \in \mathcal{T}\}$ is the "design space" for the framework $\mathcal{U}$. We need to devise a framework $\mathcal{U}$ such that the *instantaneously* $(\boldsymbol{w}_t, \alpha_t)$-fair allocations, i.e.,

$$\hat{\boldsymbol{x}}_t = \arg\max_{\boldsymbol{x} \in \mathcal{X}} U(\boldsymbol{x}; \boldsymbol{w}_t, \alpha_t), \quad \forall t \in \mathcal{T}, \tag{7}$$

are capable of achieving the desirable performance *in the long-term.* To be more specific, we will focus on the following two online performance metrics.

3.3.1 Efficiency Loss in the Regret Form. Given the utility maximization framework $\mathcal{U}$, we measure the incurred *efficiency loss* based on the cumulative regret up to the T-th slot:

$$Reg_T(\mathcal{U}) \triangleq \sum_{t=1}^{T} \mathbb{E}[f_t(\hat{\boldsymbol{x}}_t)] - \sum_{t=1}^{T} f_t(\boldsymbol{x}^\star), \tag{8}$$

where $\boldsymbol{x}^\star$ is the offline benchmark according to Definition 1. Moreover, $\{\hat{\boldsymbol{x}}_t : t \in \mathcal{T}\}$ are the decisions generated according to (7) in our framework $\mathcal{U}$.

3.3.2 ***Minimum Guarantee Violation***.
Given the utility maximization framework $\mathcal{U}$, we evaluate the cumulative violation of the minimum guarantee requirement according to

$$Vio_T(\mathcal{U}) \triangleq \sum_{m=1}^{M} \mathbb{E}\left[\sum_{t=1}^{T} g_{t,m}(\hat{\boldsymbol{x}}_t)\right]^+, \tag{9}$$

where $[\cdot]^+ = \max(\cdot, 0)$, and the expectation is taken with respect to the random variables $\{\theta_t : \forall t \in \mathcal{T}\}$.

Note that if $Reg_T(\mathcal{U})$ and $Vio_T(\mathcal{U})$ are both sub-linear in T, then the proposed framework $\mathcal{U}$ performs asymptotically as well as the offline benchmark in Definition 1. In this case, the adopted online allocations $\{\hat{\boldsymbol{x}}_t : \forall t \in \mathcal{T}\}$ are instantaneously $(\boldsymbol{w}_t, \alpha_t)$-fair, and can also ensure the allocation efficiency and the minimum guarantee requirement in the long-term. We will investigate how to devise the sequence $\{(\boldsymbol{w}_t, \alpha_t) : \forall t \in \mathcal{T}\}$ in Section 4. The key challenges are two-fold:

- The values of the loss function $f_t(\cdot)$ and the minimum guarantee requirement $\{g_{t,m}(\cdot) \le 0, \forall m \in \mathcal{M}\}$ are only disclosed sequentially at the end of each slot. This requires us to devise $(\boldsymbol{w}_t, \alpha_t)$ based only on previously revealed information via a proper learning technique.
- To overcome the above challenge, it is necessary to figure out the fundamental connection between the weighted alpha-fairness and the universal learning rationale (i.e., the exploitation-exploration tradeoff).

We will introduce how to address these challenges and propose a novel utility maximization framework in Section 4.

4 EW-ALPHA-FAIR FRAMEWORK

In this section, we propose an exponentially weighted alpha-fair (EW-Alpha-Fair) framework $\mathcal{U}$. This framework will appropriately devise 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 $Reg_T(\mathcal{U})$ and the minimum guarantee violation $Vio_T(\mathcal{U})$ are both sub-linear in T. Our study in this paper relies on the following assumption regarding the convexity and the bounded partial derivative, which are the common conditions made in related works (e.g., [20–22]).

Assumption 1. *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.,*

$$\left|\frac{\partial f_t(\boldsymbol{x})}{\partial x_n}\right| \le G \text{ and } \left|\frac{\partial g_{t,m}(\boldsymbol{x})}{\partial x_n}\right| \le G,\ \forall n, m. \tag{10}$$

We will introduce the core idea and the major components of the EW-Alpha-Fair framework $\mathcal{U}$ in Section 4.1 and Section 4.2, respectively. We then present the corresponding theoretical performance in Section 4.3. Due to space limitation, we provide the proofs in our technical report [29].

4.1 Core Idea

To figure out the connection between the weighted alpha-fairness concept and the universal learning rationale, we will investigate the impact of the weight vector $\boldsymbol{w}_t$ and the fairness level α_t on the corresponding $(\boldsymbol{w}_t, \alpha_t)$-fair allocation decision $\hat{\boldsymbol{x}}_t$ in (7). Note that (7) is a convex optimization problem, and the Karush-Kuhn-Tucker (KKT) conditions enable us to derive $\hat{\boldsymbol{x}}_t$ in a closed-form as shown in Proposition 1.

Proposition 1. *As defined in (7), when $\alpha_t > 0$, the $(\boldsymbol{w}_t, \alpha_t)$-fair allocation $\hat{\boldsymbol{x}}_t$ is given by*

$$\hat{x}_{t,n} = \frac{w_{t,n}^{1/\alpha_t}}{\sum_{i=1}^{N} w_{t,i}^{1/\alpha_t}} \cdot A, \quad \forall n \in \mathcal{N}. \tag{11}$$

The closed-form expression (11) has two-fold implications. First, the allocation amount $\hat{x}_{t,n}$ is positively related to the weight $w_{t,n}$ associated with the node n. Second, the fairness level α_t controls the magnitude of the weight vector's impact. More specifically, in the case of $\alpha_t \to 0$, all the resource will go to the node with the largest weight. In the case of $\alpha_t \to \infty$, all the nodes tend to equally share all the resource. These two implications provide us with valuable hints on how to devise a novel framework $\mathcal{U}$ and implement the learning rationale (i.e., the exploitation-exploration tradeoff) by adjusting the adopted weighted alpha-fairness.[3]

- **Exploitation:** It is necessary to adopt the merit-based weight vectors in the framework $\mathcal{U}$. That is, the weight $w_{t,n}$ should be positively related to the merit achieved by the allocation to node n in previous time slots. Such a merit should simultaneously cover the incurred loss and the minimum guarantee violation in a proper manner. Overall, the weight vectors $\{\boldsymbol{w}_t : \forall t \in \mathcal{T}\}$ correspond to the exploitation in the online allocation problem.
- **Exploration:** The aforementioned merit-based weights will gradually accumulate as the time goes by. However, overly relying on the previous information may result in the sub-optimal outcome in the future, especially when the unknown environment varies in a non-i.i.d. manner. Therefore, the framework $\mathcal{U}$ also needs to mitigate the weight vector's impact by properly increasing the allocation fairness levels as time goes by. Intuitively, a larger α_t means a higher fairness level, which reduces the weight vector's impact and leads to a stronger exploration in online allocation.

The above discussions unveil the intrinsic connection between the weighted alpha-fairness and the learning rationale (i.e., the exploitation-exploration tradeoff). Our proposed EW-Alpha-Fair framework $\mathcal{U}$ will appropriately manipulate exploitation and exploration via the weight vectors $\{\boldsymbol{w}_t : \forall t \in \mathcal{T}\}$ and the fairness levels $\{\alpha_t : \forall t \in \mathcal{T}\}$, respectively. Next let us introduce the details.

4.2 Framework Design

The above discussion indicates that the merit-based weight vectors should simultaneously cover the efficiency and the minimum guarantee. Hence we follow the previous studies (e.g., [19][30]), and consider the following Lagrangian function in each slot t

$$\mathcal{L}_t(\boldsymbol{x}_t, \boldsymbol{\mu}_t) \triangleq f_t(\boldsymbol{x}_t) + \sum_{m=1}^{M} \mu_{t,m} g_{t,m}(\boldsymbol{x}_t) - \frac{\delta_t \|\boldsymbol{\mu}_t\|_2^2}{2}, \tag{12}$$

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,

[3]The result in Proposition 1 is also discussed by [7] in the context of caching policy design. However, [7] does not consider the connection between weighted alpha-fairness and exploration-exploitation tradeoff, which is the key insight in this paper.

$\mu_{t,m}$ 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).

Next, we introduce the four major components in our proposed EW-Alpha-Fair framework $\mathcal{U}$.

4.2.1 Weighted Alpha-Fair Allocation. In each slot t, the EW-Alpha-Fair framework $\mathcal{U}$ will adopt the $(\boldsymbol{w}_t, \alpha_t)$-fair allocation $\hat{\boldsymbol{x}}_t$ defined in (7). 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,

4.2.2 Increasing Fairness Level. The EW-Alpha-Fair framework $\mathcal{U}$ will adopt the following fairness levels

$$\alpha_t \triangleq t^{\epsilon} G \sqrt{\frac{1+M}{2\ln(N)}}, \quad \forall t \in \mathcal{T}. \tag{13}$$

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 4.3, it determines the three-way tradeoff among fairness level, efficiency loss, and minimum guarantee violation. Furthermore, (13) only requires limited prior knowledge on the derivative bound G, the number of minimum guarantee requirements M, and the number of network nodes N.

4.2.3 Merit-Based Weight. The EW-Alpha-Fair framework $\mathcal{U}$ will adopt the merit-based weight based on the Lagrangian function (12). Intuitively, minimizing the Lagrangian function implicitly reduces the incurred 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

$$w_{t+1,n} = w_{t,n} \cdot \exp\left(-\frac{\partial \mathcal{L}_t(\hat{\boldsymbol{x}}_t; \boldsymbol{\mu}_t)}{\partial x_n}\right), \ \forall n \in \mathcal{N}, \tag{14}$$

where $\hat{\boldsymbol{x}}_t$ is the adopted allocation in slot t. Note that the updating rule in (14) is based on an exponential function, which is inspired by the classic Hedge algorithm [31]. Our study in this paper is not to improve this algorithm or its variants, but to generalize the weighted alpha-fair utility to the online setting.

4.2.4 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). The EW-Alpha-Fair framework $\mathcal{U}$ will update the Lagrangian multipliers $\boldsymbol{\mu}_{t+1}$ according to

$$\begin{aligned}\mu_{t+1,m} &= \left[\mu_{t,m} + \beta_t \cdot \frac{\partial \mathcal{L}_t(\hat{\boldsymbol{x}}_t; \boldsymbol{\mu}_t)}{\partial \mu_m}\right]^+, \\ &= \left[\mu_{t,m} + \beta_t \cdot \Big(g_{t,m}(\hat{\boldsymbol{x}}_t) - \delta_t \mu_{t,m}\Big)\right]^+,\end{aligned} \tag{15}$$

where the second equality follows directly from the definition in (12). Moreover, β_t is a step-size parameter, and δ_t is the regularization parameter. The EW-Alpha-Fair framework $\mathcal{U}$ will determine the two parameters β_t and δ_t based on the current fairness level α_t according to

$$\delta_t = \frac{3(1+M)G^2 A}{\alpha_t}, \tag{16a}$$

$$\beta_t = \frac{1}{(t+1)\delta_t}. \tag{16b}$$

The above expressions are determined with the goal of reducing the cumulative regret $Reg_T(\mathcal{U})$ and the violation $Vio_T(\mathcal{U})$ achieved by our proposed framework $\mathcal{U}$. We refer interesting readers to our technical report [29].

Algorithm 1: EW-Alpha-Fair Framework $\mathcal{U}$

```
  Input  : $A$, $N$, $M$, and $G$
  Output : Allocation decisions $\hat{\boldsymbol{x}}_t$ for each slot $t$.
1 Determine $\epsilon \in (0,1)$
2 Initialize $\boldsymbol{w}_1 = \boldsymbol{1}_N$ and $\boldsymbol{\mu}_1 = \boldsymbol{0}_M$
3 for $t = 1$ to $T$ do
      /* At the beginning of this slot */
4     Calculate the fairness level $\alpha_t$ according to (13).
5     Adopt the $(\boldsymbol{w}_t, \alpha_t)$-fair allocation decision $\hat{\boldsymbol{x}}_t$ according
      to Proposition 1.
      /* At the end of this slot */
6     Observe the revealed loss function $f_t(\cdot)$ and the
      minimum guarantee requirements $\{g_{t,m}(\cdot) : \forall m \in \mathcal{M}\}$.
7     Update the weight vector $\boldsymbol{w}_{t+1}$ according to (14).
8     Update $\boldsymbol{\mu}_{t+1}$ according to (15).
```

Algorithm 1 summarizes the EW-Alpha-Fair framework $\mathcal{U}$. First of all, we specify the parameter $\epsilon \in (0,1)$ in Line 1. 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 the following two phases:

- *Lines 4-5:* At the beginning of slot t, we calculate the fairness level α_t according to (13), and then adopt the $(\boldsymbol{w}_t, \alpha_t)$-fair allocation $\hat{\boldsymbol{x}}_t$ according to Proposition 1.
- *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. Accordingly, we calculate the weight vector $\boldsymbol{w}_{t+1}$ and Lagrangian multipliers $\boldsymbol{\mu}_{t+1}$ for the next slot according to (14) and (15), respectively.

Before analyzing the theoretical performance, let us highlight three advantages of the EW-Alpha-Fair framework. First, Algorithm 1 is adaptive in time and does not require the prior knowledge on the total number of slots T. Hence the operator does not need to perform the doubling-trick (e.g., [21]). Second, Algorithm 1 requires very limited prior knowledge. Specifically, (13)-(16) only require the number of nodes N, the number of constraints M, the resource amount A, and the derivative bound G. Third, Algorithm 1 has a low computational complexity and does not involve high-dimensional projection operation. The above three aspects correspond to our first remark for the EW-Alpha-Fair framework $\mathcal{U}$.

Remark 1. *The EW-Alpha-Fair framework in Algorithm 1 is an adaptive online allocation framework, which requires limited prior knowledge and has a low computation complexity.*

4.3 Performance Analysis

Now we analyze the theoretical performance achieved by the EW-Alpha-Fair framework $\mathcal{U}$. As we will show in Theorem 1, this framework achieves a sub-linear efficiency loss and a sub-linear minimum guarantee violation. The major analysis for this result is to show that the cumulative regret $Reg_T(\mathcal{U})$ and violation $Vio_T(\mathcal{U})$ satisfy the following inequality

$$\begin{aligned}&Reg_T(\mathcal{U}) + \frac{\left[Vio_T(\mathcal{U})\right]^2}{2M\left[\frac{1}{\beta_1} - \delta_1 + J_T(\boldsymbol{\delta})\right]} \leq \\ &\left[\alpha_T \ln(N) + \frac{(1+M)G^2 J_T(\boldsymbol{\alpha})}{2}\right] A + MNA^2G^2 J_T(\boldsymbol{\beta}),\end{aligned} \tag{17}$$

where $J_T(\boldsymbol{\alpha})$, $J_T(\boldsymbol{\beta})$, and $J_T(\boldsymbol{\delta})$ are given by

$$J_T(\boldsymbol{\alpha}) \triangleq \sum_{t=1}^{T} \frac{1}{\alpha_t},\ \ J_T(\boldsymbol{\beta}) \triangleq \sum_{t=1}^{T} \beta_t,\ \ J_T(\boldsymbol{\delta}) \triangleq \sum_{t=1}^{T} \delta_t. \tag{18}$$

Note that the three formulas in (18) may scale in T. Nevertheless, the inequality $\sum_{t=1}^{T} \frac{1}{t^z} \leq \frac{T^{1-z}}{1-z}, \forall z \in (0,1)$ implies that all the three formulas have sub-linear upper bounds. Based on this observation, rearranging (17) leads to our main result in Theorem 1.

Theorem 1. *For any $\epsilon \in (0,1)$, the EW-Alpha-Fair framework $\mathcal{U}$ in Algorithm 1 achieves the following efficiency loss in the regret form and minimum guarantee violation*

$$\begin{aligned}&Reg_T(\mathcal{U}) \leq AG \cdot \Omega_T(\epsilon),\\ &Vio_T(\mathcal{U}) \leq AG\sqrt{\frac{24MCT^{2-\epsilon}}{1-\epsilon}\left[\sqrt{2N} + \frac{\Omega_T(\epsilon)}{T}\right]},\end{aligned} \tag{19}$$

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

$$\Omega_T(\epsilon) \triangleq \left(T^{\epsilon} + \frac{T^{1-\epsilon}}{1-\epsilon}\right) \cdot C + \frac{T^{\epsilon}}{\epsilon} \cdot \frac{MN}{6C}, \tag{20}$$

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

Theorem 1 shows that the EW-Alpha-Fair 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 1. This is our second remark for the proposed framework.

Remark 2. *The EW-Alpha-Fair framework in Algorithm 1 ensures the asymptotic efficiency and minimum guarantee by adopting the weighted alpha-fair allocations instantaneously in each time slot.*

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

- *Fairness Level:* Recall that the EW-Alpha-Fair framework $\mathcal{U}$ adopts the increasing fairness levels. Specifically, (13) indicates 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 1 indicates that the 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 $\mathcal{U}$ adopts less fair allocations, which lead to insufficient exploration (i.e., excessive exploitation). When ϵ is increasing to 1, our framework $\mathcal{U}$ adopts more fair allocations, which lead to the excessive exploration (i.e., insufficient exploitation).
- *Minimum Guarantee Violation:* Theorem 1 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 EW-Alpha-Fair framework $\mathcal{U}$.

Remark 3. *The EW-Alpha-Fair 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:*

$$O\left(T^{\epsilon}\right),\ O\left(\tfrac{T^{1-\epsilon}}{1-\epsilon} + \tfrac{T^{\epsilon}}{\epsilon}\right),\ O\left(\sqrt{\tfrac{T^{2-\epsilon}}{1-\epsilon}}\right). \tag{21}$$

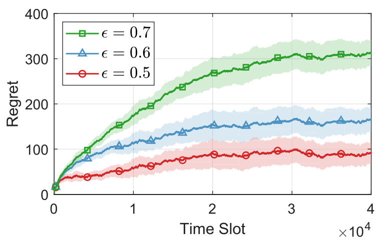
5 NUMERICAL RESULTS

In this section, we demonstrate how the EW-Alpha-Fair framework functions in two networking applications, i.e., size-based scheduling and file caching.

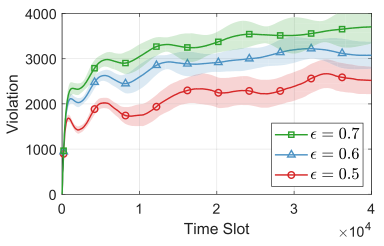
5.1 Size-Based Scheduling

5.1.1 Background. Congestion control is a critical issue in the modern data-center network. Most active congestion control schemes are partly implemented in the network switches. The bandwidth allocation (or the scheduling policy) installed on the switches 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 (e.g., [32]). 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 based on the online allocation framework.

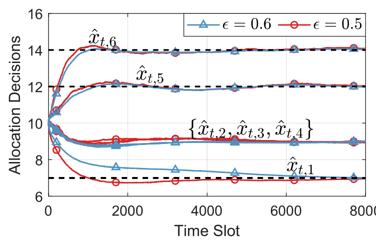
5.1.2 Formulation. We consider a data-center network with a set $\mathcal{N} = \{1, 2, ..., N\}$ of N classes of flows. The N classes of flows are



(a) Efficiency loss in the regret form



(b) Minimum guarantee violation



(c) Allocation decisions $\{\hat{\boldsymbol{x}}_t : \forall t\}$

Figure 1: Numerical results in size-based scheduling

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.

Efficiency Measurement: The size-based scheduling aims to prioritize the short flows and reduce the average FCT. For this goal, we follow NUMFabric [33] and consider the following loss function

$$f_t(\boldsymbol{x}_t) = -\sum_{n=1}^{N} \frac{x_{t,n}}{s_{t,n}}, \tag{22}$$

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 (22) 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 (22) is in favor of reducing the average FCT [33].

Minimum Guarantee Requirement: The loss function in (22) implicitly induces the bandwidth allocation outcome that prioritizes short flows. To prevent the starvation of long flows, we take into account a total of N minimum guarantee requirements $\{g_n(\boldsymbol{x}) \leq 0, \forall n \in \mathcal{N}\}$, where $g_n(\boldsymbol{x})$ is

$$g_n(\boldsymbol{x}) = \rho_n - x_n. \tag{23}$$

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 implicitly avoid flow starvation in size-based scheduling.

5.1.3 Numerical Results. We consider $N = 6$ classes of flows and randomly generate $\boldsymbol{s}_t$ according to the heavy-tail Pareto distribution with mean values $[60, 50, 50, 50, 50, 50]$. That is, the class-1 flows are longer than other classes on average (which is unknown in advance). Moreover, we suppose that the total bandwidth is $A =$ 60Gbps, 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 EW-Alpha-Fair framework $\mathcal{U}$ to the size-based scheduling problem under different parameters $\epsilon = \{0.5, 0.6, 0.7\}$. Fig. 1 shows the average results of multiple simulation runs.

Fig. 1(a) and Fig. 1(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. 1(a) and Fig. 1(b), a greater fairness increasing speed results in larger efficiency loss and more violation. Furthermore, the fluctuation of curves in Fig. 1(b) is due to the back-and-forth movement of the allocation decisions around the minimum guarantee thresholds $\boldsymbol{\rho}$ (as shown in Fig. 1(c)).

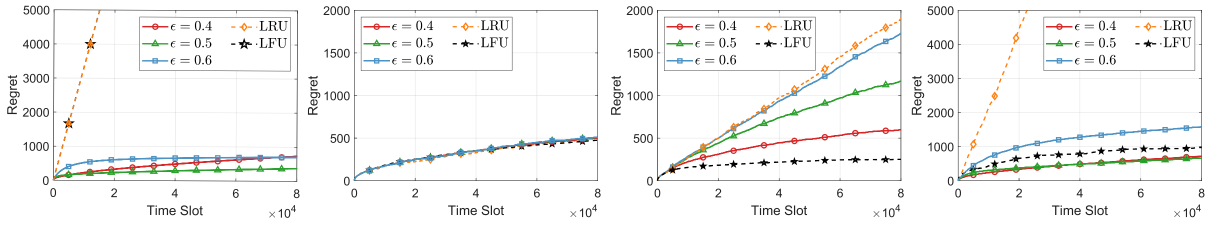
Fig. 1(c) plots the allocation decisions $\{\hat{\boldsymbol{x}}_t : \forall t \in \mathcal{T}\}$ for cases $\epsilon \in \{0.5, 0.6\}$. In the first slot, the N classes of flows equally share the bandwidth for both cases. During the remaining slots, the two cases are slightly different:

- 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. 1(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 $\epsilon = 0.5$ responses to the minimum guarantee requirement faster than the case $\epsilon = 0.6$.

To sum up, the EW-Alpha-Fair framework enables the network switches to prioritize short flows and avoid flow starvation without prior flow size information.

5.2 Caching Capacity Allocation

5.2.1 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. 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. 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



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

Figure 2: File caching results (The solid curves represent EW-Alpha-Fair framework with $\epsilon \in \{0.4, 0.5, 0.6\}$)

caching policies, such as Least-Recently-Used (LRU) and Least-Frequently-Used (LFU). Next we demonstrate how to apply our EW-Alpha-Fair framework to the file caching problem.

5.2.2 Formulation. We consider a file library with a set $\mathcal{N} = \{1, 2, ..., N\}$ 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. 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

$$f_t(\boldsymbol{x}_t) \triangleq -\sum_{n=1}^{N} \sigma_{t,n} v_n \min\left(\frac{x_{t,n}}{b_n}, 1\right), \tag{24}$$

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. We will use this sub-gradient in the merit-based weight updating (14).

Note that minimizing the loss function in (24) 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 framework. To have a reasonable comparison, we will not model the minimum guarantee, since LFU and LRU do not consider this.

5.2.3 Numerical Results. We compare the EW-Alpha-Fair 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. 2 plots the average results of multiple simulation runs under the caching capacity $A = 10$. The four sub-figures correspond to different file request characteristics.

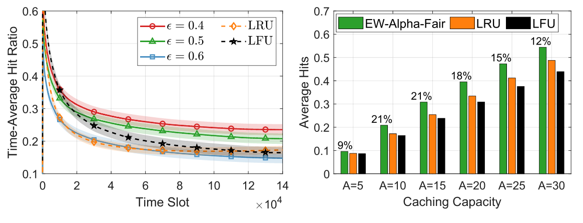
Periodic Request: Fig. 2(a) plots the results under the periodic file requests, i.e., $\{1, 2, ..., 30, 1, 2, ..., 30, ...\}$. Previous studies (e.g., Lemma 3.7 in [35]) have shown that LRU and LFU cannot achieve a sub-linear regret under this type of periodic request. As shown in Fig. 2(a), the two dash curves increase linearly. Furthermore, the other three solid curves in Fig. 2(a) correspond to our EW-Alpha-Fair 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 1, 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. 2(b) and Fig. 2(c) focus on the stationary file request with the uniform popularity and Zipf popularity, respectively. In general, LFU performs optimally in response to the stationary requests, since it estimates the file popularity based on the observed request frequency.

- In Fig. 2(b), the five curves almost overlap with each other. This means that both the EW-Alpha-Fair framework and the LRU caching policy can perform as well as LFU when the files are equally popular.
- In Fig. 2(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 different from the regret bound in Theorem 1. The reason is that Theorem 1 focuses on the non-stationary environment, and the corresponding regret bound may not be tight in the stationary case.

Mixed Request: Fig. 2(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 EW-Alpha-Fair framework could outperform LFU in this case.

The above file request patterns are kind of artificial. Now we evaluate the caching performance based on the shot noise request model [36], which captures the ephemeral YouTube video requests. Fig. 3 shows the results, where the vertical axis in the two sub-figures represents the time-average hits. Fig. 3(a) plots the results under the caching capacity $A = 10$. In this case, the EW-Alpha-Fair 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. 3(b) plots the results achieved by the EW-Alpha-Fair framework with $\epsilon = 0.5$, LRU, and LFU under various caching capacities. The percentage values at the top of green bars (i.e., EW-Alpha-Fair) represent the



(a) Case of $A = 10$ (b) Impact of caching capacity

Figure 3: Results of shot noise request model

improvement compared to the orange bars (i.e., LRU). Overall, the EW-Alpha-Fair framework improves the average hits up to 21% compared to the state-of-the-art caching policies.

6 CONCLUSION AND FUTURE WORK

In this paper, we develop a novel utility maximization framework with fairness guarantee for online resource allocation. This framework will ensure the asymptotic efficiency by adopting the utility-maximizing allocation. The major challenge of designing this framework lies in the coupling of the desired fairness guarantee and the unknown allocation efficiency. We extend the classic weighted alpha-fair utility from the perspective of merit-based weight and the increasing fairness level. The two aspects implicitly integrate the weighted alpha-fairness with the learning rationale. Under this framework, the utility-maximizing allocation in each time slot is weighted alpha-fair. Moreover, our framework performs asymptotically as well as the offline optimal/efficient outcome. We also demonstrate the advantages of this framework in the size-based scheduling and file caching problems. This paper focuses on the divisible resource allocation. It would be interesting to consider how to ensure the asymptotic performance of allocating indivisible resource via weighted alpha-fairness.

ACKNOWLEDGMENTS

This work was supported in part by National Key R&D Program (NKRDP) of China under Grant 2019YFB1802803, in part by The Chinese University of Hong Kong (CUHK) under Grant 6905407, and in part by the RGC SRFS2122-4S02.

REFERENCES

[1] Frank P Kelly, Aman K Maulloo, and David KH Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 49(3):237–252, 1998.
[2] Daniel Pérez Palomar and Mung Chiang. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, 24(8):1439–1451, 2006.
[3] Mung Chiang, Steven H Low, A Robert Calderbank, and John C Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, 2007.
[4] Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on networking*, 8(5):556–567, 2000.
[5] Tian Lan, David Kao, Mung Chiang, and Ashutosh Sabharwal. An axiomatic theory of fairness in network resource allocation. *IEEE INFOCOM*, 2010.
[6] Carlee Joe-Wong, Soumya Sen, Tian Lan, and Mung Chiang. Multiresource allocation: Fairness–efficiency tradeoffs in a unifying framework. *IEEE/ACM Transactions on Networking*, 21(6):1785–1798, 2013.
[7] Mostafa Dehghan, Laurent MassouliÃľ, Don Towsley, Daniel Sadoc MenaschÃľ, and Y. C. Tay. A utility optimization approach to network cache design. *IEEE/ACM Transactions on Networking*, 27(3):1013–1027, 2019.
[8] Eduardo Hargreaves, Claudio Agosti, Daniel MenaschÃľ, Giovanni Neglia, Alexandre Reiffers-Masson, and Eitan Altman. Fairness in online social network timelines: Measurements, models and mechanism design. *Performance Evaluation*, 129:15–39, 2019.
[9] John F Nash Jr. The bargaining problem. *Econometrica: Journal of the econometric society*, pages 155–162, 1950.
[10] Ehud Kalai and Meir Smorodinsky. Other solutions to nash's bargaining problem. *Econometrica: Journal of the Econometric Society*, pages 513–518, 1975.
[11] Georgios S Paschos, Apostolos Destounis, and George Iosifidis. Online convex optimization for caching networks. *IEEE/ACM Transactions on Networking*, 28(2):625–638, 2020.
[12] Tareq Si Salem, Giovanni Neglia, and Stratis Ioannidis. No-regret caching via online mirror descent. *IEEE ICC*, 2021.
[13] Shangxing Wang, Hanpeng Liu, Pedro Henrique Gomes, and Bhaskar Krishnamachari. Deep reinforcement learning for dynamic multichannel access in wireless networks. *IEEE Transactions on Cognitive Communications and Networking*, 4(2):257–265, 2018.
[14] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S Mirrokni, and Cliff Stein. Online stochastic packing applied to display ad allocation. In *European Symposium on Algorithms*, pages 182–194. Springer, 2010.
[15] Tianyi Chen, Qing Ling, and Georgios B Giannakis. An online convex optimization approach to proactive network resource allocation. *IEEE Transactions on Signal Processing*, 65(24):6350–6364, 2017.
[16] Fengjiao Li, Jia Liu, and Bo Ji. Combinatorial sleeping bandits with fairness constraints. *IEEE Transactions on Network Science and Engineering*, 7(3):1799–1813, 2019.
[17] Mehrdad Mahdavi, Tianbao Yang, and Rong Jin. Online decision making under stochastic constraints. *NIPS workshop on Discrete Optimization in Machine Learning*, 2012.
[18] Hao Yu, Michael J Neely, and Xiaohan Wei. Online convex optimization with stochastic constraints. *NeurIPS*, 2017.
[19] Kechao Cai, Xutong Liu, Yu-Zhen Janice Chen, and John CS Lui. An online learning approach to network application optimization with guarantee. *IEEE INFOCOM*, 2018.
[20] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.
[21] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011.
[22] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325, 2016.
[23] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. A survey of serendipity in recommender systems. *Knowledge-Based Systems*, 111:180–192, 2016.
[24] Laura Schelenz. *Diversity-Aware Recommendations for Social Justice? Exploring User Diversity and Fairness in Recommender Systems*, page 404âĂŞ410. Association for Computing Machinery, New York, NY, USA, 2021.
[25] Adam Wagstaff. Qalys and the equity-efficiency trade-off. *Journal of health economics*, 10(1):21–41, 1991.
[26] Yaozhong Wu, Christoph H Loch, and Ludo Van der Heyden. A model of fair process and its limits. *Manufacturing & Service Operations Management*, 10(4):637–653, 2008.
[27] Thomas Vossen, Michael Ball, Robert Hoffman, and Michael Wambsganss. A general approach to equity in traffic flow management and its application to mitigating exemption bias in ground delay programs. *Air Traffic Control Quarterly*, 11(4):277–292, 2003.
[28] Shie Mannor, John N Tsitsiklis, and Jia Yuan Yu. Online learning with sample path constraints. *Journal of Machine Learning Research*, 10(3), 2009.
[29] Technical report. https://wang-zhi-yuan.github.io/file/TR_MobiHoc22.pdf.
[30] Rodolphe Jenatton, Jim Huang, and Cédric Archambeau. Adaptive algorithms for online convex optimization with long-term constraints. *ICML*, 2016.
[31] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *IEEE Annual Foundations of Computer Science*, 1995.
[32] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. pfabric: Minimal near-optimal datacenter transport. *ACM SIGCOMM*, 2013.
[33] Kanthi Nagaraj, Dinesh Bharadia, Hongzi Mao, Sandeep Chinchali, Mohammad Alizadeh, and Sachin Katti. Numfabric: Fast and flexible bandwidth allocation in datacenters. *ACM SIGCOMM*, 2016.
[34] Guoqiang Zhang, Yang Li, and Tao Lin. Caching in information centric networking: A survey. *Computer networks*, 57(16):3128–3141, 2013.
[35] Georgios Paschos, George Iosifidis, Giuseppe Caire, et al. Cache optimization models and algorithms. *Foundations and Trends® in Communications and Information Theory*, 16(3-4):156–343, 2020.
[36] Stefano Traverso, Mohamed Ahmed, Michele Garetto, Paolo Giaccone, Emilio Leonardi, and Saverio Niccolini. Temporal locality in today's content caching: why it matters and how to model it. *ACM SIGCOMM Computer Communication Review*, 43(5):5–12, 2013.