IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 28, 2020, accepted May 5, 2020, date of publication May 22, 2020, date of current version June 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2996630

Online Multi-Resource Social Welfare Maximization for Non-Preemptive Jobs

CHAOQUN YOU[1], (Member, IEEE), CHENG REN[2], AND LEMIN LI[1]
[1]The Key Laboratory of Optical Fiber Sensing and Communications, University of Electronic Science and Technology of China, Chengdu 611731, China
[2]School of Electrical Engineering and Information, Southwest Petroleum University, Chengdu 610500, China

Corresponding author: Chaoqun You (chaoqunyou@gmail.com)

This work was supported in part by the NSFC Fund under Grant 61671130, in part by the SWPU of Science and Technology Higher Learning Innovation Ability Enhancement, China under Grant 2019CXTD06, and in part by the Collaboration Project of University and Nanchong, China under Grant 19SXHZ0018.

ABSTRACT Multi-resource allocation is ubiquitous in datacenters. Usually, the datacenter scheduler associates each user with a utility function and then allocates multiple resources so as to maximize the social welfare, which is the sum of the utilities of the users. We refer to this problem as the Social Welfare Maximization (SWM) problem. Meanwhile, given the dynamic nature of datacenters, the scheduler is supposed to support non-preemptive scheduling, where jobs of users that are being processed would not be interrupted by the reconfiguration of resources. Non-preemptive solutions to SWM incorporating pricing considerations include dynamic pricing and bidding/auctions. However, both strategies are quite complex. Meanwhile, traditional Lagrange solutions to SWM does not support non-preemptive job scheduling. In this paper, we propose Adaptive Dominant Resource Fairness (ADRF), that provides near-optimal social welfare where each user is assigned an α-fair utility function and when $1 < \alpha \leq 2$. Based on ADRF, we propose Cumulative ADRF (C-ADRF), an online algorithm that greedily assigns the idle resources to the current poorest user, making it quite simple to schedule jobs from users non-preemptively. Compared with the polynomial or exponential form of time complexities in other SWM solutions, C-ADRF requires only $O(\log N)$ work, where N is the number of users. Extensive simulations using Google and Facebook cluster-data traces show an impressive behavior of C-ADRF in providing social welfare, only 3% and 2% away from the optimal social welfare, respectively.

INDEX TERMS Multi-resource allocation, fairness, social welfare, online scheduling, non-preemption.

I. INTRODUCTION

The problem of multi-resource allocation in a cloud datacenter, which is naturally constrained in terms of CPU time, memory, communication links, and other resources, is of broad practical and theoretical interest. As the datacenter workloads surge, providing fine-grained multi-resource allocation among users, each of which submits multiple jobs, has become increasingly important. Usually, schedulers solve such problems by associating each user with a utility function and then provide an allocation that maximizes the social welfare, which is the sum of all users' utilities [1]–[6]. We refer to this allocation policy as the Social Welfare Maximization (SWM) problem.

We consider the *non-preemptive* job scheduling in multi-resource SWM. That is, once a job starts its service, its ongoing process cannot be preempted (paused) or migrated due to the reconfiguration of resource allocation. This is because preemptions or migrations require storing the state of interrupted jobs and recovering them at a later time, which is operational costly. In general, current data-intensive computing frameworks in datacenters, such as Hadoop Fair Scheduler [7] and Mesos [8], do not support such migrations or preemptions [9]. Instead, given the dynamic nature of datacenters, the scheduler is required to make *online* decisions on which job to allocate resources to only if ongoing jobs finish and resources free up, without interfering the existing resource configurations.

The conventional solution obtained by solving the multi-resource SWM directly using the Lagrange method [10], however, is a preemptive scheduler, *i.e.*, every time a user arrives or departures, the scheduler needs to recompute the SWM problem, and ongoing jobs may be preempted due to the reconfigurations of resources. More common

The associate editor coordinating the review of this manuscript and approving it for publication was Fan-Hsun Tseng.

non-preemptive solutions incorporating pricing considerations include dynamic pricing [11]–[13] and bidding/auction [4]–[6], [14] algorithms. However, these strategies can be quite complex [11], especially in data-intensive frameworks in cloud datacenters.

In this paper, we propose a new multi-resource allocation algorithm, Adaptive Dominant Resource Fairness (ADRF), that provides near-optimal social welfare where each user is assigned an α-fair utility function and when $1 < \alpha \leq 2$. Then, based on ADRF, we propose Cumulative Adaptive Dominant Resource Fairness (C-ADRF), an online non-preemptive scheduler that will eventually converge to ADRF.

ADRF is designed based on Dominant Resource Fairness (DRF) [15], a compelling multi-resource fair allocation policy that is simple enough to achieve online using the well-known progressive filling algorithm [16]. Progressive filling greedily allocates idle resources to the user with the current lowest resource share (*i.e.*, the poorest user). Therefore, progressive filling is the fundamental idea to implement ADRF online. Despite the advantages offered by progressive filling, how well can ADRF approximate to the optimal social welfare remains *unclear*. In this paper, we study the upper bound of the gap of social welfare between ADRF and SWM and prove that when $1 < \alpha \leq 2$, this upper bound can be minimized.

Moreover, based on ADRF, we propose an online algorithm, C-ADRF, that can eventually converge to the static ADRF solutions to provide non-preemptive job scheduling. As we mentioned in the last paragraph, progressive filling is used to implement ADRF online. However, a naive application of progressive filling to ADRF leads to the starvation of certain users, that the user with the lowest current resource share may not be served if its resource demand exceeds the available resource capacities. To avoid this shortcoming, C-ADRF is designed to cumulate freed up resources until the idle resources could support the next job from the current poorest user, and then to the current second poorest user, and so on.

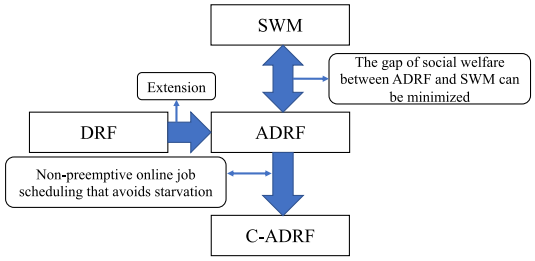


FIGURE 1. System model.

The outline of the system model is shown in Fig. 1 and the contributions of this paper are four-fold:

- We propose ADRF, a multi-resource fair allocation policy that provides near-optimal social welfare. (Section IV-C)
- We mathematically study the gap between the social welfare ADRF provides and the optimal social welfare in SWM. We upper bound the gap and prove that when $1 < \alpha \leq 2$, the upper bound of the gap can be minimized. (Section IV-D)
- We propose a greedy online scheduling algorithm, C-ADRF, based on the progressive filling algorithm to execute the offline ADRF in practice. C-ADRF avoids the starvation caused by the naive application of progressive filling. (Section V)
- We conduct extensive simulations using Google and Facebook cluster-data traces. Simulation results show that the online allocation solutions of C-ADRF perform well, only around 3% and 2% differences from the optimal solutions in Google and Facebook traces, respectively. (Section VI)

The rest of this paper is structured as follows: Section II introduces the related work. Section III describes the system model and the optimization objective. Section IV introduces ADRF and mathematically describe the upper bound of the gap between the social welfare ADRF provides and the optimal social welfare. In Section V, we propose C-ADRF, an online scheduler of ADRF, which provides dynamic fair multi-resource scheduling when jobs are indivisible and non-preemptive. We study the performance of ADRF and C-ADRF separately in Section VI. Section VII concludes this paper.

II. RELATED WORK

In this section, we start with the introduction of dynamic provisioning policies for SWM (Section II-A) Next we introduce the existing multi-resource fair sharing works in the datacenter networks (Section II-B).

A. DYNAMIC PROVISIONING FOR SOCIAL WELFARE MAXIMIZATION

Many works incorporate pricing considerations to design online mechanisms for resource allocation so as to maximize the provider's revenue [11], [14], [17]–[20] or social welfare [1]–[6], [12], [13], [20]. Given the natural dynamic requirements of job scheduling, both dynamic pricing and bidding/auctions have been proposed as major solutions in most online scheduling schemes. Dynamic pricing schemes [11]–[13] are provider strategies to better cope with unpredictable user demand: pricing should be leveraged to influence demand to generate more revenue or social welfare. Therefore, to implement dynamic pricing, the resource provider needs to collect demand information and empirically predict the future demand arrivals. Bidding and auctions [4]–[6], [14], [20]–[24], on the other hand, are strategies jointly considering user-provider interactions: users can use the resource instances only if their bids exceed the instance price given by the provider. Although both dynamic pricing and auctions/bidding can efficiently allocate resources

dynamically, their strategies can be quite complex, especially in data-intensive frameworks in cloud computing.

Cloud datacenters, in which data-intensive computing frameworks such as MapReduce [25], Pregel [26] and Dryad [27] gathered, need to make scheduling decisions at high speeds. Typical datacenter schedulers, including Hadoop Fair Scheduler [7], YARN [28] and Mesos [8], make thousands of scheduling decisions per second. Therefore, besides dynamic pricing and bidding/auctions, a simple and efficient online algorithm still remains an open problem. In this paper we propose a greedy online algorithm in this paper that is easy to implement in practice.

Meanwhile, the study of SWM to dynamically allocate a single resource (e.g., bandwidth allocation) is widely adopted in flow and congestion control problem [29]. For the single resource case, the SWM problem is also referred to as the well-known Network Utility Maximization (NUM) problem. In NUM, each user is associated with a utility function and the optimization objective is to maximize the sum of the utility functions of all users. The α-fair utility function is used in NUM, and the limit $\alpha \to +\infty$ gives max-min fairness. Max-min fairness can be easily achieved in a greedy online manner using the well-known progressive filling algorithm [16] when jobs from users can be infinitely divisible. This method can be easily extended to the case when jobs are indivisible and non-preemptive to be scheduled [9]. We construct a similar optimization objective but relate it to multiple resources. Multi-resource allocation for SWM is considered in [30], but the utility function of each user is simple, restricting its formulation to the dominant share of each user. Other works considering the dynamic multi-resource allocation using max-min fairness [31]–[33], as we claimed in Section II-A, mainly concern about the fairness among users rather than the social welfare maximization, and is therefore orthogonal to our work.

B. MAX-MIN FAIR ALLOCATION

Despite the extensive computing system literature on max-min fair resource allocation, many of the existing works limit their discussions to the allocation of a single resource type [29], [34]. Max-min fair allocation can be achieved using the well known progressive filling algorithm [16]. Progressive filling increased all users' shares at the same rate until the user requiring the least resources has been fully satisfied, then the second, and so on. In other words, max-min fairness can be achieved greedily online by always assigning idle resources to the current user with the lowest share.

A compelling work, DRF [15], extends max-min fairness to multiple resource types. DRF attempts to execute max-min fairness on the *dominant shares* of all users. DRF can also be achieved by the progressive filling algorithm [9]. DRF has quickly attracted significant attentions and has been generalized to many dimensions [30]–[33], [35], [36]. However, all these consider the multi-resource fair sharing in the cloud from a purely operational perspective. On the other hand, [30] extends DRF from the perspective of economics.

TABLE 1. Notations.

$\mathcal{A}$	the set of users
N	the number of users
$\mathcal{R}$	the set of resources
M	the number of resource types
c_r	the resource capacity of resource r
D_{ir}	the ratio between the demand of user i on resource r and c_r
d_{ir}	the normalized resource demand of user i on resource r
x_i	the dominant resource share of user i
$U(x_i)$	the utility of user i with a dominant share of x_i
$W_i(x_i)$	the weight utility function of user i with a dominant share of x_i

However, this paper only considers the social welfare performance as a simple summation of each user's dominant share. Reference [35] captures the trade-off between allocation fairness and efficiency but their solution still does not consider the non-preemption of jobs in online scheduling systems. All these multi-resource fair allocation algorithms perform poorly in the maximization of social welfare.

III. SYSTEM MODEL

In this section, we first introduce the notations that would be used in the paper (Section III-A). Then we introduce the formulation of SWM (Section III-B). Next we interpret the inefficiencies of conventional solutions to SWM using the Lagrange method (Section III-C). Finally, we introduce the requirement of online scheduling in practice (Section III-D).

A. MULTI-RESOURCE JOBS AND DOMINANT SHARE

In datacenter environments, users usually submit *fine-grained* jobs to data-intensive computing frameworks like MapReduce [25] and Spark [37], each of which requires a slice of a server for a short amount of time that is not known to the scheduler in advance [8].

Denote the set of users by $\mathcal{A} = \{1, 2, \ldots, N\}$ and each user has multiple identical jobs to be assigned to the data center, and the set of hardware resource by $\mathcal{R}$, Let $M = |\mathcal{R}|$. Here the resources can be CPU, memory, storage, etc. Let $\mathcal{C} = \{c_1, \ldots, c_M\}$ be the resource capacity of resource r in the datacenter. Without loss of generality, we normalize the total availability of every resource to 1, *i.e.*, $c_r = 1$ for all resources $r \in R$. Let D_{ir} be the ratio between the demand of user i for resource r and c_r. For all users i and resources r, we denote $d_{ir} = \frac{D_{ir}}{\max_{r'} D_{ir'}}$; we refer to these demands as normalized user demands. For simplicity, we assume positive demands, *i.e.*, $d_{ir} > 0$ for all user i and resource r. The reason we use the normalized user demands is that the capacity of every single resource is also normalized to 1. The *dominant resource* of user i is the resource for which the user's job requires the largest d_{ir}. A user's *dominant share* is defined as the fraction of the dominant resource that the user has been allocated, which we denote as x_i. The utiltiy of a user i depends on its dominant share x_i. Meanwhile, user i is allocated $d_{ir}x_i$ fraction, and we can express the capacity constraint on resource r as follows,

$$\sum_{i \in \mathcal{A}} d_{ir} x_i \leq 1. \tag{1}$$

For example, consider a cluster with 9 CPUs and 18 GB RAM shared between two users. User 1 demands ⟨1CPU, 4GB⟩ per (divisible) job, and user 2 demands ⟨3CPU, 1GB⟩ per (divisible) job. When CPU is resource 1 and RAM is resource 2, $D_{11} = 1/9, D_{12} = 2/9, D_{21} = 1/3$ and $D_{22} = 1/18$. According to the definition of d_{ir}, we have $d_{11} = 1/2, d_{12} = 1, d_{21} = 1$ and $d_{22} = 1/6$. In this example, the dominant resource of user 1 is the memory as each of its jobs demands 1/9 of the total CPU and 2/9 of the total memory. On the other hand, the dominant resource of user 2 is CPU, as each of its jobs requires 1/3 of the total CPU and 1/18 of the total memory.

B. OPTIMIZATION OBJECTIVE

Each user derives a certain utility $U(x_i)$ when getting the dominant share x_i. Since a user's utility increases with the share allocated, we follow the economic principle of diminishing marginal utility and assume that $U(x_i)$ is a strictly increasing, concave function.

The optimization objective of this paper is to maximize the social welfare, which is formulated as follows.

$$\begin{aligned} &\max_{x_i} \sum_{i\in\mathcal{A}} U(x_i) \\ &s.t. \sum_{i\in\mathcal{A}} d_{ir}x_i \le 1, \quad \forall r \in \mathcal{R}, \\ &\quad x_i \ge 0, \quad \forall i \in \mathcal{A}. \end{aligned} \tag{2}$$

The first constraint represents the resource capacity constraints, that the total allocation to all the users on a certain resource type r should not exceed the capacity of this resource provided by the datacenter. The second constraint means that each user's dominant share is larger or equal to zero. We denote this optimization problem as the SWM problem.

In SWM, the utility function can be interpreted as imposing different notions of fair resource allocation to users [29]. Famous forms of fairness such as *proportional fairness* and *max-min fairness* can be unified by considering utility functions of the form

$$U(x_i) = \frac{x_i^{1-\alpha}}{1-\alpha}, \tag{3}$$

for some $\alpha > 0$. Resource allocation using the above utility function is called α-fair. It has been shown in [29] that proportional fair resource allocation is achieved when $\alpha \to 1$, at this point the utility function is equivalent to the logarithmic function, that is, $U(x_i) = \log x_i$. Meanwhile, the limit $\alpha \to +\infty$ gives DRF, which is an extension of max-min fairness from a single resource to multiple resources.

C. INEFFICIENCY IN CONVENTIONAL SOLUTIONS

We first summarize the standard Lagrange method to solve SWM, which will provide design motivation on the pricing aspect of our proposed solution. We note that this method always schedules users and jobs in a preemptive manner.

The Lagrangian of SWM is defined as follows,

$$L(x,h) = \sum_{i\in\mathcal{A}} U(x_i) - \sum_{r\in\mathcal{R}} h_r\Big(\sum_{i\in\mathcal{A}} d_{ir}x_i - 1\Big) \tag{4}$$

where h_r is the Lagrange multiplier of SWM corresponding to resource r. When SWM achieves its optimal solution, the first derivative of the Lagrangian equals to zero. Setting $\partial L(x,h)/\partial x_i = 0$ for each i gives

$$U'(x_i) = \sum_{r\in\mathcal{R}} h_r d_{ir} \tag{5}$$

This is used to derive the Lagrange dual problem, which is then solved to determine the optimal h_r values and the corresponding optimal x_i values. If we consider h_r as the virtual unit price of resource r, then $\sum_{r\in R} h_r d_{ir}$ can be interpreted as the price of user i in consuming one job with demand $d_{ir}, r \in R$.

To solve SWM online using the Lagrange method, the allocation x_i or the Lagrange multiplier h_r is based on the demands $d_{ir}, r \in R$, of user i that currently exists in the system. Whenever a user arrives or departs, the allocation is recomputed. We might have to reconfigure all users to obtain the configuration specified by the new solution. This could include preempting jobs that are being processed and replacing them by the others due to the changes in the resource allocation decision. Jobs that are thus preempted come at an undesirable cost when they resume computation from where they stopped [7], [38]. Typical datacenter schedulers, such as Hadoop Fair Scheduler [7] and Mesos [8], do not support such expensive computations as they make thousands of scheduling decisions per second along with user arrivals and departures.

D. ONLINE JOB ARRIVAL AND DEPARTURE

We consider the *online* scenario where users and their jobs arrive dynamically, with unknown arrival times a priori. Conventional solutions to the SWM problem is usually in the *offline* setting, where all users and jobs are available for scheduling at the beginning. Furthermore, such offline solutions often treat jobs preemptively: whenever a user is added or deleted, the allocation configuration is recomputed and users that are preempted due to the change of allocation are unreasonable assumed to be able to resume computation from where they stopped without any cost.

In practical dynamic datacenter environments, however, offline solutions are neither practical nor efficient. This is because: (1) the datacenter scheduler needs to make frequent allocation decisions whenever a new user is added or when resources free up. Typical datacenter schedulers, including Hadoop Fair Scheduler [7] and Mesos [8], make thousands of decisions per second as jobs continuously finish and their resources need to be allocated to new jobs. Therefore, offline solutions to SWM require recomputations of every user every time a user is added or deleted, leading to an expensive recomputation cost; (2) the resumption of preempted jobs will

cause a large amount of overhead; (3) datacenter jobs are non-preemptive as they are *indivisible* and have to be executed as whole entities. Therefore, in this paper, we propose a greedy online algorithm such that it is fairly simple and efficient to schedule datacenter jobs in practice.

IV. ADAPTIVE DOMINANT RESOURCE FAIRNESS

In this section, we first provide an overview of the algorithm we propose, Adaptive Dominant Resource Fairness, ADRF. Then it is necessary to review the definition of DRF and its general form GW-DRF. Next we introduce the method to determine the particular form that can provide near-optimal social welfare in GW-DRF and name it ADRF. At last we study the social welfare difference between ADRF and SWM, mathematically describing the upper bound of the gap.

A. OVERVIEW

ADRF provides an near-optimal solution to SWM when $1 < \alpha \leq 2$, where each user is assigned an α-fair utility function. The design process can be clearer using the flowchart shown in Fig. 2. Inspired by the Lagrangian method that solves SWM, a pricing scheme shown in Equation (5) can be obtained when the first derivative of the Lagrangian equals to zero. However, this pricing scheme does not discriminate the dominant resource from other resource types for each user. Therefore, we propose the max-min fair pricing scheme and prove it leads to a weighted generalized DRF (GW-DRF) with a specific form of the weight function, which we name ADRF.

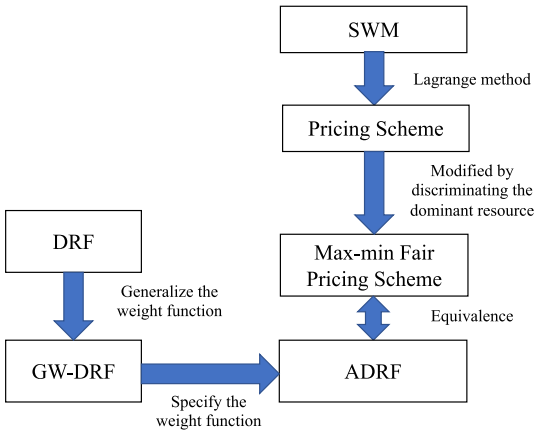


FIGURE 2. Flowchart in designing ADRF.

B. REVIEW: FORMAL DEFINITION OF DRF

DRF extends the max-min fairness from single resources case to multi-resource cases. DRF achieves max-min fairness on the dominant shares of each user. That is, the dominant shares of all users are equalized in DRF. The formal definition of DRF is given as follows.

Definition 1 (DRF): We call a multi-resource allocation $(x_1, x_2, \ldots, x_N)$ feasible when the capacity constraint shown in (1) is satisfied. Given the dominant share x_i for each user i, we say that a feasible allocation satisfies DRF when, for each user i, any increase in the dominant share x_i would cause a decrease in the dominant share $x_{i'}$ of some user i' satisfying $x_{i'} \leq x_i$.

C. GW-DRF AND ADRF

We introduce ADRF in order to overcome the obstacles in SWM shown in Section III-C, which will facilitate our design of the online scheduling algorithm C-ADRF as shown in Section V. Moreover, ADRF is a specially designed case of the Generalized Weighted DRF (GW-DRF). GW-DRF is a generalization of the original DRF by assigning a generalized weight function to users. The formal definition of GW-DRF is given as follows.

Definition 2 (GW-DRF): Given some weight function $W_i(x_i)$ for each user i, we say that a feasible allocation satisfies GW-DRF when, for each user i, any increase in the dominant share x_i would cause a decrease in the dominant share $x_{i'}$ of some user i' satisfying $W_{i'}(x_{i'}) \leq W_i(x_i)$.

In essence, this generalization allows flexible assignment of some individual utility value to x_i for each user i in our consideration of multi-resource fairness. Note that the original definition of DRF is recovered with

$$W_i(x_i) = x_i, \quad i \in A.$$

Just as it was suggested in (5), there exists a close relationship between the resource allocation policy in SWM and the pricing scheme. Since the limit $\alpha \to \infty$ gives max-min fairness on the dominant resource (*i.e.*, DRF), we could reasonably indicate that there also exists a particular pricing scheme for GW-DRF as GW-DRF is a generalization of DRF. Next we introduce a weighted max-min pricing scheme that is closely related to the resource allocation in GW-DRF.

As it was shown in Section IV-A, the unit resource prices are the same for all uses in the pricing scheme of (5). This fails to discriminate the dominant resource from other resource types for each user. Moreover, the conventional pricing scheme based on (5) fails to support datacenter jobs non-preemptively. Therefore, instead of (5), we propose to use $U'(x_i) = \beta_i q_i$, where q_i is a virtual unit price of the dominant resource of user i and β_i is a pre-determined constant. That is, given β_i, the virtual per-job price of user i is determined only by the per-unit price q_i of the dominant resource of user i. Here $\beta_i q_i$ can be viewed as an approximation to the right-hand side of (5), where q_i provides a rough representation of $\{h_r\}$ in (5), with user-dependent adjustments based on its dominant share, while β_i aggregates the resource demand $\{d_{ir}\}$ of user i. We will discuss the optimal choices of β_i in Section IV-C.

Now we formally define the pricing scheme for GW-DRF. We consider a max-min fair pricing scheme, where the worst treated users (*i.e.*, the users who are charged the highest weighted per-unit price on its dominant resource) is charged as low a price as possible.

Definition 3 (Max-Min Fair Pricing): We identify a pricing scheme $\mathbf{q} = (q_1, q_2, \ldots, q_N)$ to be the per-unit price vector on the dominant resource of the N users. Since $U'(x_i) = \beta_i q_i$, corresponding to (1), we say that an allocation under

price vector $\mathbf{q}$ is feasible when, for every resource type $r \in R$, we have

$$\sum_{i\in\mathcal{A}} U'^{-1}(\beta_i q_i) d_{ir} \leq 1.$$

We call a price vector *max-min fair* when the corresponding allocation is feasible, and when it is not possible to decrease q_i without losing feasibility or increasing $q_{i'}$ of another user i' with $q_{i'} \geq q_i$

We next show how the weighted max-min pricing scheme leads to a weighted DRF with specific form of the weight function.

Proposition 4: Consider a dominant resource price vector $\mathbf{q} = (q_1, q_2, \ldots, q_N)$ and the corresponding dominant share vector $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, *i.e.*, for $i = 1, 2, \ldots, N$, we have $U'(x_i) = \beta_i q_i$. Then the pricing scheme is max-min fair if and only if the dominant resource share $(x_1, x_2, \ldots, x_N)$ satisfies the GW-DRF with weight functions $W_i(x_i) = \frac{\beta_i}{U'(x_i)}$.

Proposition 4 is trivial as $W_i(x_i) = \frac{1}{q_i}$, which can be obtained through $U'(x_i) = \beta_i q_i$ and $W_i(x_i) = \frac{\beta_i}{U'(x_i)}$. It provides us one possible form of weight function in GW-DRF. We refer to this specific form of GW-DRF as ADRF. The formal definition of the ADRF allocation scheme as follows.

Definition 5 (ADRF): An allocation policy satisfies Adaptive Dominant Resource fairness (ADRF) if the weight function in the GW-DRF is $W_i(x_i) = \frac{\beta_i}{U'(x_i)}$. The dominant share assigned to a user x_i in ADRF is referred to as *adaptive dominant share*.

D. OPTIMIZING β_I

In the previous subsection, we develop a fairness criteria for pricing schemes to connect the price-taking based service with the concept of max-min fairness and then get one possible formulation of $W_i(x_i)$. However, it remains to determine suitable values β_i for all users, in order to maximize the social welfare. We can decrease the gap between the social welfare ADRF provides and the optimal social welfare through the adjustment of the value of β_i. In this way, the objective in determining the value of β_i is to minimize the gap.

As it was shown in [29], max-min fairness can be achieved in SWM when $\alpha \to \infty$. Therefore, an alternative expression of the GW-DRF is shown as follows.

$$\begin{aligned} &\max_{x_i} \lim_{\xi\to\infty} \sum_{i\in\mathcal{A}} \frac{W_i^{1-\xi}(x_i)}{1-\xi} \\ &s.t. \sum_{i\in\mathcal{A}} d_{ir} x_i \leq 1, \quad \forall r \in \mathcal{R}, \end{aligned} \tag{6}$$

where $\xi \to \infty$. Here we use ξ instead of α to distinguish it from the α of the utility function used in $W_i(x_i)$. The Lagrangian of problem (6) is given by

$$L(x,p) = \lim_{x_i\to\infty} \sum_{i\in\mathcal{A}} \frac{W_i^{1-\xi}(x_i)}{1-\xi} - \sum_{r\in\mathcal{R}} p_r \Big(\sum_{i\in\mathcal{A}} d_{ir} x_i - 1\Big), \tag{7}$$

where p_r is the Lagrangian multiplier of (6) for the resource r capacity constraint. Setting $\partial L(x,p)/\partial x_i = 0$ for each i gives

$$\lim_{x_i\to\infty} W_i^{-\xi}(x_i) W_i'(x_i) = \sum_{r\in\mathcal{R}} p_r d_{ir}. \tag{8}$$

Further, the KKT conditions require that

$$p_r \Big(\sum_{i\in\mathcal{A}} x_i d_{ir} - 1\Big) = 0 \quad \text{and} \quad p_i \geq 0. \tag{9}$$

Denote the optimal result of (6) as $\hat{x}_i$. As long as $\hat{x}_i$ is reached, at least one of the resources in the system must be saturated. We assume that only one of the resources r_1 is saturated. This is without loss of generality because of the heterogeneity of resource capacities and user demands, rendering it unlikely to have two or more simultaneously saturated resources. Thus,

$$\sum_{i\in\mathcal{A}} \hat{x}_i d_{ir_1} = 1, \tag{10}$$

and p_{r_1} is the only non-zero Lagrange multiplier. Furthermore, considering (3), we have $U'(x_i) = x_i^{-\alpha}$. Hence, $W_i(x_i) = \beta_i x_i^{\alpha}$. At this point, (8) can be rewritten as

$$\hat{x}_i = \lim_{x_i\to\infty} \Big(\frac{p_{r_1} d_{ir_1}}{\alpha \beta_i^{1-\xi}}\Big)^{\frac{1}{-\alpha\xi+\alpha-1}}. \tag{11}$$

Let (11) substitute for the $\hat{x}_i$ in (10), we have

$$p_{r_1} = \lim_{x_i\to\infty} \frac{1}{\sum_{i\in\mathcal{A}} \big(\frac{d_{ir_1}}{\alpha\beta_i^{1-\xi}}\big)^{\frac{1}{-\alpha\xi+\alpha-1}} d_{ir_1}}, \tag{12}$$

Substituting for the p_{r_1} from (11), as $\xi \to \infty$, we obtain

$$\hat{x}_i = \frac{1}{\beta_i^{\frac{1}{\alpha}} \sum_{j\in\mathcal{A}} d_{jr_1} \beta_j^{-\frac{1}{\alpha}}}. \tag{13}$$

The same calculation can be used in problem (2), denote the optimal solution as x_i^* and the resource saturated as r_2 in problem (2) respectively, then we can obtain

$$x_i^* = \frac{1}{d_{ir_2}^{\frac{1}{\alpha}} \sum_{j\in\mathcal{A}} d_{jr_2}^{1-\frac{1}{\alpha}}}. \tag{14}$$

Denote GAP as the gap between the social welfare ADRF provides and the optimal social welfare in SWM, then we have

$$\begin{aligned} \texttt{GAP} &= \sum_{i\in\mathcal{A}} U(x_i^*) - \sum_{i\in\mathcal{A}} U(\hat{x}_i) && \text{(15a)} \\ &= \sum_{i\in\mathcal{A}} \frac{(d_{ir_2}^{\frac{1}{\alpha}} \sum_{j\in\mathcal{A}} d_{jr_2}^{1-\frac{1}{\alpha}})^{\alpha-1}}{1-\alpha} \\ &\quad - \sum_{i\in\mathcal{A}} \frac{(\beta_i^{\frac{1}{\alpha}} \sum_{j\in\mathcal{A}} d_{jr_1} \beta_j^{-\frac{1}{\alpha}})^{\alpha-1}}{1-\alpha} && \text{(15b)} \end{aligned}$$

Although we hope that GAP could be as small as possible, we can not determine the saturated resource types r_1 and r_2 in ADRF and SWM. Moreover, the design of ADRF should not

depend on which type of resource is saturated. Denote $d_{ir_{\max}}$ as the maximum demand of user i and $d_{ir_{\min}}$ as the minimum demand of user i. We then analyze the upper bound of GAP in the following three cases.

- When $0 < \alpha < 1$, we have

$$\mathrm{GAP} \le \sum_{i\in\mathcal{A}} \frac{(d_{ir_{\min}}^{\frac{1}{\alpha}} \sum_{j\in\mathcal{A}} d_{jr_{\max}}^{1-\frac{1}{\alpha}})^{\alpha-1}}{1-\alpha} - \sum_{i\in\mathcal{A}} \frac{(\beta_i^{\frac{1}{\alpha}} \sum_{j\in\mathcal{A}} d_{jr_{\max}} \beta_j^{-\frac{1}{\alpha}})^{\alpha-1}}{1-\alpha}. \quad (16)$$

- When $\alpha > 1$, we have

$$\mathrm{GAP} \le \sum_{i\in\mathcal{A}} \frac{(d_{ir_{\min}}^{\frac{1}{\alpha}} \sum_{j\in\mathcal{A}} d_{jr_{\min}}^{1-\frac{1}{\alpha}})^{\alpha-1}}{1-\alpha} - \sum_{i\in\mathcal{A}} \frac{(\beta_i^{\frac{1}{\alpha}} \sum_{j\in\mathcal{A}} d_{jr_{\max}} \beta_j^{-\frac{1}{\alpha}})^{\alpha-1}}{1-\alpha}. \quad (17)$$

In the next step, we design ADRF according to the upper bound of GAP shown in the right hand side of (16) and (17) and hope to find a β_i that can minimize the accordingly upper bound.

Before we define the optimal value of β_i for ADRF, we firstly introduce the following lemma and its proof is shown in the appendix due to space concern.

Lemma 6: Consider a positive semi-definite matrix H_n, in which for each individual item we have $h_{ii} \ge 0$, $h_{ij} \le 0$ $(i \ne j)$. If for another matrix H_n' the following

$$\begin{cases} h'_{ii} & \ge h_{ii} \ge 0, \\ h_{ij} & \le h'_{ij} \le 0 \quad (i \ne j), \end{cases} \quad (18)$$

holds, then H_n' is also a positive semi-definite matrix.

With this lemma, the following proposition can be proved. The proof of the following proposition is also shown in the appendix.

Proposition 7: ADRF can provide the smallest worst-case GAP (*i.e.* the upper bound of GAP) if $\beta_i = d_{ir_{\max}}$, and the upper bound of GAP can be minimized when $1 < \alpha \le 2$. More specifically, the upper bound of GAP is

$$\sum_{i\in\mathcal{A}} \frac{(d_{ir_{\min}}^{\frac{1}{\alpha}} \sum_{j\in\mathcal{A}} d_{jr_{\min}}^{1-\frac{1}{\alpha}})^{\alpha-1} - (d_{ir_{\max}}^{\frac{1}{\alpha}} \sum_{j\in\mathcal{A}} d_{jr_{\max}}^{1-\frac{1}{\alpha}})^{\alpha-1}}{1-\alpha}. \quad (19)$$

Proposition 7 shows that GAP can be strictly upper bounded when $1 < \alpha \le 2$, which indicates that ADRF can provide near-optimal social welfare. So far, ADRF is still an offline algorithm, the whole section defines ADRF and computes its distance to the optimal solutions in SWM. Based on ADRF, in the next section we will proceed to design its online form.

V. CUMULATIVE ADAPTIVE DOMINANT RESOURCE FAIRNESS

So far, our design of ADRF is still a static solution that provides a near-optimal social welfare. As we mentioned in the beginning, our goal has always been the non-preemptive manner of online job scheduling. Therefore, ADRF only is not enough.

Fortunately, there is a well known greedy online algorithm, progressive filling [16], that can achieve DRF, which is basically the fundamental idea in designing the online version of ADRF.

In this section, we start by considering a naive extension of progressive filling to implement ADRF online, which is named Naive ADRF for convenience in later use (Section V-A). However, Naive ADRF may lead to starvations of certain upcoming users. Then we introduce our approach, Cumulative ADRF (C-ADRF), that overcomes this defect (Section V-B). Finally, we analyze the time complexity of C-ADRF (Section V-C) and consider how well does C-ADRF approximate ADRF (Section V-D).

A. INEFFICIENCIES OF THE NAIVE ADRF

Progressive filling is an idealized algorithm to achieve DRF online by always assigning idle resources to the user with the lowest current dominant share whose constraints the resource satisfies [15]. Therefore, progressive filling allocates resources in a greedy online manner, without preempting any jobs that are being processed [9].

Naive ADRF uses progressive filling directly by allocating the idle resources to the user with the lowest current adaptive dominant share whenever a resource frees up. Specifically, when a job finishes its service on a server, its resources free up. The scheduler will then determine if the freed up resources can accommodate the next job from the user with the least adaptive dominant share. If the idle resources can not support a job from the poorest user, the Naive ADRF scheduler will proceed to the next user with the second least adaptive dominant share, and so on.

However, this policy has an obvious drawback — it might starve some users waiting in line. Consider a datacenter with only two users and one server with resources ⟨9 CPUs, 18 GB memory⟩. Suppose that currently three jobs from user 1 are being processed, each of which requires ⟨3 CPUs, 1 GB memory⟩, and a newly joined user 2 requires ⟨1 CPUs, 2 GB memory⟩. Only when two jobs from user 1 free up their resources can one job from user 2 be scheduled. However, if the jobs from user 1 finish one by one, every time a job leaves, the Naive ADRF scheduler will continue to serve user 1's next job since the freed up resources can not accommodate one job from user 2. This leads to the starvation of user 2 and its fair allocation will never be achieved.

B. C-ADRF

To address the user starvation problem in *Naive ADRF*, we propose Cumulative ADRF (C-ADRF), that always

assigns freed up resources to the current poorest user, but does not proceed to the current second poorest user if the idle resources are not enough to support the next job from the current poorest user. Instead, C-ADRF will cumulate freed up resources until it is enough to serve the next job from the current poorest user.

For the same example used in Section V-A, when user 2 arrives, it is the poorest user as its adaptive dominant resource share is zero. At this point, if only one job from user 1 finishes, C-ADRF will stop and wait until the next job from user 1 leaves and the cumulated freed up resources are enough to support the first job from user 2.

Algorithm 1 C-ADRF Pseudo-Code

procedure JobCompletion(H_i)
$W(x_i) \leftarrow W(x_i - \mu_i)$
while data center is not full **do**
  Let k be the user with the least adaptive dominant share;
  if user k's job, H_k, can be accommodated by the idle resources **then**
    Assign H_k to the data center;
    Update $W(x_k) \leftarrow W(x_k + \mu_k)$
  else
    wait until another job $H_j, j \in A$, to be finished;
    procedure JobCompletion (H_j)
  end if
end while

The pseudo-code of C-ADRF is shown in Algorithm 1. Every time a job of a certain user leaves the system, Algorithm 1 is triggered. If there are enough idle resources to support one queueing job H_k from the user with the least adaptive dominant share k, C-ADRF schedules H_k next and then updates the adaptive dominant share of user k. On the contrary, there are not enough idle resources to accommodate a queueing job from the poorest user k, C-ADRF will wait until another job H_j from user j to leave and cumulate the freed up resources. C-ADRF then repeats the above process until the queueing job H_k from the poorest user k can be scheduled using the cumulative idle resources.

C. TIME COMPLEXITY OF C-ADRF

Proposition 8: The time complexity of C-ADRF is $O(\log N)$, where N is the number of users that currently exists in the datacenter.

Proof: C-ADRF does not change the greedy nature of progressive filling. It always assigns the idle resources to the current poorest user. Therefore, the time complexities of C-ADRF and progressive filling are the same. Progressive filling needs to sort the dominant shares of all the N users that currently exist in the datacenter and requires $O(\log N)$ time complexity per job [15], [39]. This analysis also applies to C-ADRF. C-ADRF needs to sort the adaptive dominant shares of all the N users that currently exist in the datacener and requires $O(\log N)$ time complexity per job. ■

The time complexity of problem (2) solving by barrier method is $O(\varphi N^3)$ [10], where φ is the number of iterations related with the number of constraints M in SWM. Only when $\alpha \rightarrow 1$ can φ have a linear relationship with M. In other cases, it is deeply hard to mathematically analyze the number of iterations, but φ will be a number much larger than it would be in the case when $\alpha \rightarrow 1$. Therefore, the time complexity of C-ADRF is much smaller than the time complexity in solving SWM directly. Considering the indivisible and non-preemptive natures of users' jobs and the decrease in time complexity, C-ADRF trades some social welfare for the improvement in implement efficiency.

D. C-ADRF CONVERGENCE PROPERTY

How well does C-ADRF approximate ADRF? Given that C-ADRF returns an allocation vector with an integer number of jobs to be scheduled, we modify ADRF in problem (6) with an additional integer constraint. The following proposition shows that C-ADRF will eventually converge to ADRF allocation with an integer constraint.

Proposition 9: Consider a datacenter with N users, no matter when the ith user ($i = 1, 2, \ldots, N$) arrives, the C-ADRF algorithm will converge to the allocation of ADRF with integer constraints.

Proof: Progress filling achieves max-min fairness, even when there exit integer constraints on the allocation vector. Therefore, given ADRF with integer constraints, the progressive filling approach will make it keep serving the poorest user with the least $W_i(x_i)$. On the other hand, in C-ADRF, the cumulative feature guarantees that the poorest user is always the first to be served. Therefore, the sequence of jobs served by C-ADRF is exactly the same as the progressive filling execution of ADRF with constraints. Thus C-ADRF will eventually converge to ADRF allocation with integer constraints. ■

VI. SIMULATIONS

We have evaluated the performance of ADRF and C-ADRF separately in Matlab via extensive simulations driven by cluster-usage traces from Google [40] and Facebook [7]. The total number of lines in our Matlab files exceeds 3500. The demands of each user fit our model with two resources. We start with the ADRF simulation to testify its effectiveness in providing near-optimal social welfare (Section VI-A). We then evaluate the performance of C-ADRF, comparing it with the existing fair sharing policies in large clusters. (Section VI-B).

A. ADRF PERFORMANCE

We first compare the social welfare ADRF provides with the optimal social welfare in SWM.

1) SETUP

Every time we pick out 10 to 80 users randomly from the traces of Google and Facebook, respectively. We extract the computing demand information — the required amount of

resources — and use it as the demand input of the two algorithms for evaluation. We repeat this process for 100 times to obtain the *average* social welfare of ADRF and SWM and the worst-case estimated average social welfare of ADRF (*i.e.*, the optimal social welfare minus the upper bound of the gap).

2) RESULTS AND ANALYSIS

Fig. 3 shows the average social welfare of ADRF, SWM, and the worst-case estimated ADRF when $\alpha \to 1$ in the utility function (*i.e.*, $U(x_i) = \log x_i$). We observe that the actual average social welfare ADRF provides approximates to the optimal social welfare very well. Furthermore, the average ratio between the actual gap and the optimal social welfare is 2.19% in Google traces and 1.88% in Facebook traces, the average ratio between the upper bound and the optimal social welfare is 37.71% in Google traces and 32.05% in Facebook traces. This indicates ADRF provides social welfare fairly close the optimal welfare, but the upper bound of the gap can only provide a coarse estimation of the gap.

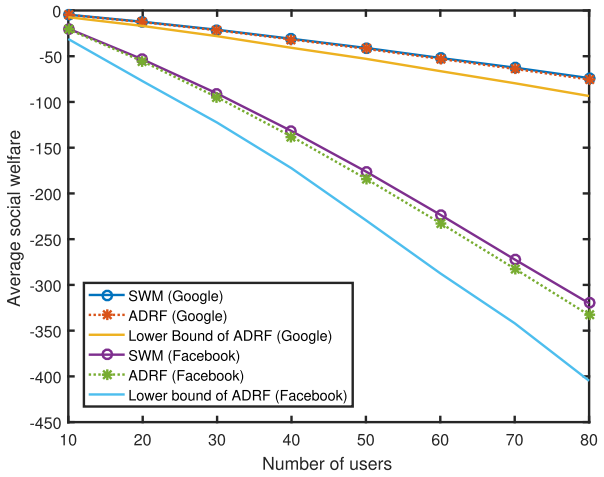


FIGURE 3. Average social welfare of ADRF and SWM when $\alpha \to 1$.

As the upper bound of the gap between the social welfare of ADRF and SWM can be minimized when $1 < \alpha \leq 2$ using the α-fair utility function, we pick several values from 1 to 2 in Table 2 to check the social welfare performance of ADRF using different utility functions. From Table 2 we can see that ADRF behaves pretty well in providing social welfare for various values of α.

TABLE 2. Average social welfare with different α in SWM and ADRF.

α	$\to 1$	1.5	2
SWM (Google)	-41	-144	-459
ADRF (Google)	-42	-147	-479
SWM (Facebook)	-176	-649	-2.17×10^3
ADRF (Facebook)	-184	-663	-2.23×10^3

B. C-ADRF PERFORMANCE

1) SETUP

In order to emulate the performance of C-ADRF, we compare C-ADRF with FIFO and five multi-resource fair online sharing policies based on max-min fairness: naive ADRF, weighted Max-Min Fairness (WMMF) [41] w.r.t. CPU share (hereafter CPU), WMMF w.r.t. memory share (hereafter Mem), generalized WMMF w.r.t. CPU share (hereafter G-CPU), WMMF w.r.t. memory share (hereafter Mem) and generalized WMMF w.r.t. memory share (hereafter G-Mem). WMMF is one of the most popular policies proposed for fair sharing in single resource allocation and can be easily extended to the multi-resource environment with respect to each type of resources. WMMF w.r.t. CPU and memory share are two simulators using constant weight 1 for each user, while generalized WMMF w.r.t. CPU and memory share are two simulators using weight function $W_i(x_i)$ for each user. Meanwhile, for the input workloads, we have sampled the jobs submitted within a 1-hour interval in the Google and Facebook traces.

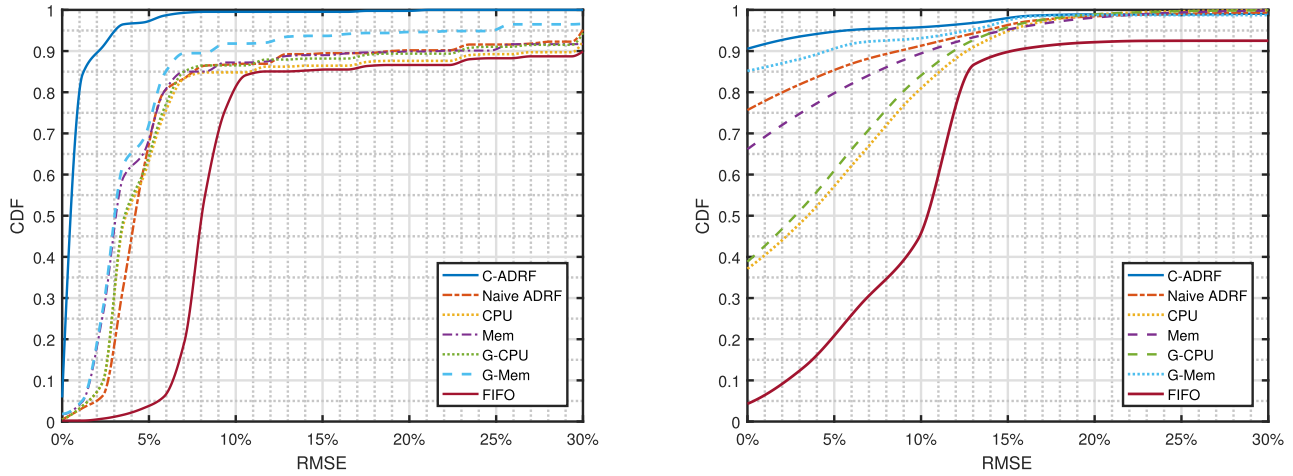
In order to evaluate the social welfare performance of C-ADRF, we measure the differences in allocation vectors between C-ADRF and SWM rather than the social welfare directly. This is because C-ADRF is a non-preemptive algorithm, the newly arrival user may have to wait in the queue to be scheduled and its instantaneous adaptive dominant share will be equal to zero. This will lead to a negative infinity utility when $U(x_i) = \log x_i$.

We measure the CDF of the Root Mean Square Error (RMSE) between the allocation vectors of SWM and one of the online scheduling algorithms (*i.e.*, C-ADRF, Naive-ADRF, CPU, G-CPU, Mem and G-Mem). That is, if an online scheduler obtains an allocation vector $\langle x_1, x_2, \ldots, x_N \rangle$ at a particular time t, and SWM provides $\langle \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N \rangle$ for the same workload, the RSME is $\sqrt{\frac{1}{N}((\frac{x_1-\hat{x}_1}{\hat{x}_1})^2 + \cdots + (\frac{x_N-\hat{x}_N}{\hat{x}_N})^2)}$. We sample the system status every 700 seconds to record the allocation vector $\langle x_1, x_2, \ldots, x_N \rangle$, users that are currently existing in the system and their resource demands, correspondingly. The optimal allocation vector can then be computed using this information in each sampled time slot.

2) RESULTS AND ANALYSIS

Fig. 4 and Fig. 5 show the CDF of RMSE using Google and Facebook traces with two different values of α in the utility function. The basic observation is that the average RMSE of C-ADRF is around 3% using Google traces and 2% using Facebook traces. That is, C-ADRF differs from the optimal solutions by only 3% in the Google cluster and 2% in the Facebook cluster! This indicates that C-ADRF behaves perfectly in providing optimal social welfare.

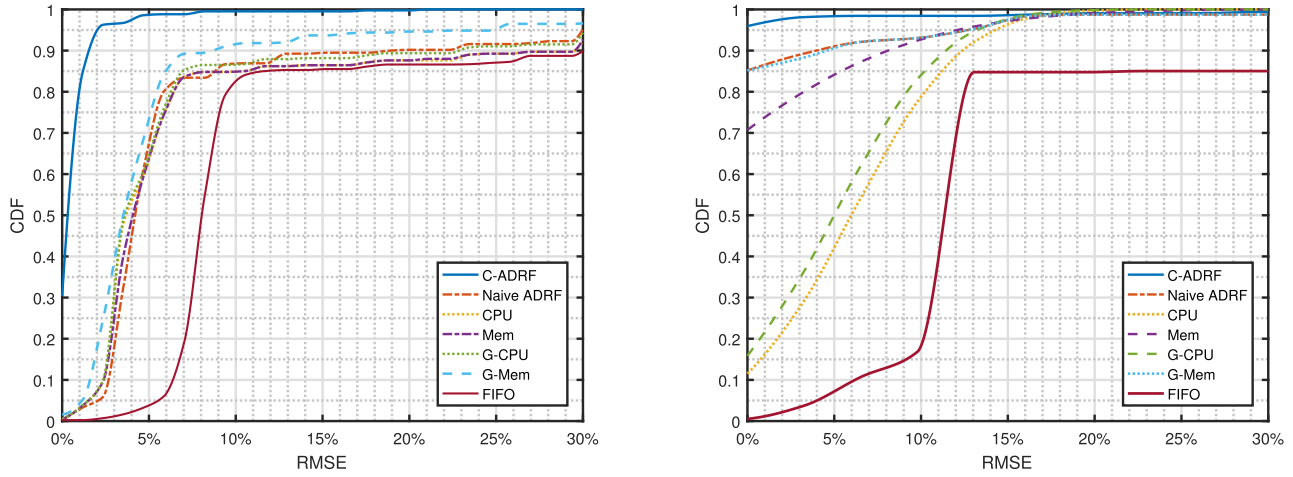
Not only that, we observe that the fair scheduling algorithms outperform FIFO unanimously, improving the social welfare of about 85% of the users by up to 10× in Google traces and about 95% of the users by up to 15× in Facebook traces. Such a significant social welfare gain derives from the design of max-min fairness to maximize social welfare. On the other hand, we note that the fair scheduling algorithms except C-ADRF experience similar social



(a) RMSE of the online algorithms in Google traces

(b) RMSE of the online algorithms in Facebook trace

FIGURE 4. Social welfare performance of C-ADRF vs. other 6 online algorithms when $\alpha \to 1$.



(a) RMSE of the online algorithms in Google traces

(b) RMSE of the online algorithms in Facebook traces

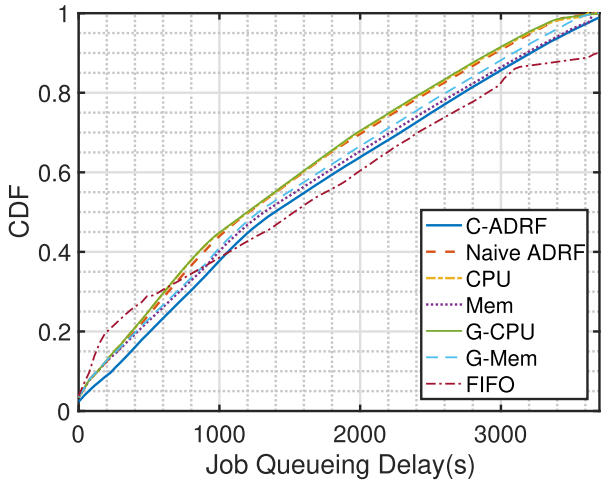
FIGURE 5. Social welfare performance of C-ADRF vs. other 6 online algorithms when $\alpha = 1.5$.

welfare performance, although G-CPU and G-Mem perform slightly better than the CPU and Mem correspondingly. This is because the weight function is designed to approximate the optimal social welfare. To our surprise, Naive ADRF behaves no better than some of the WMMF algorithms in which max-min fairness is performed only on a fixed resource. We attribute this phenomenon to the possibility of users to suffer starvation. Due to the heterogeneity of user demands for multiple resources, in Naive ADRF, users are easier to exhibit starvation. This is because only when its demand on each of the resource types is smaller than the available capacity of the corresponding resource type in the cluster, then its job can be scheduled. However, WMMF algorithms concern only one single resource, as long as a user's demand on this particular resource is smaller than the available capacity of this resource type, this user's next job can be scheduled. Therefore, users in WMMF are much less possible to be starved.

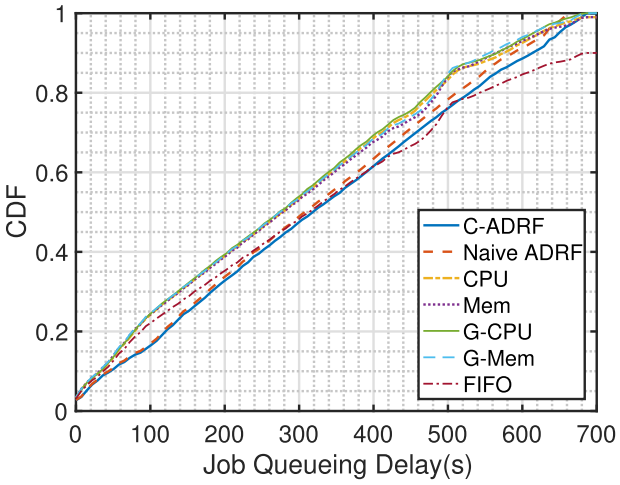
Most importantly, we can see that C-ADRF outperforms all the other algorithms significantly in providing social welfare. Since there is no starvation in C-ADRF, it can better approximate the behavior of ADRF, which is approved in Section VI-A to provide social welfare fairly close to the optimal solution.

We next evaluate the *job queueing delay*, defined for each job as the wait time from job submission to scheduling, using different algorithms. We compare the job queueing delay performance of C-ADRF with other schedulers to see if the nice social welfare gain of C-ADRF is achieved at the expense of significant job queueing delay.

Fig. 6 and Fig. 7 show the CDF of job queueing delay. We note that jobs experienced similar queueing delays across

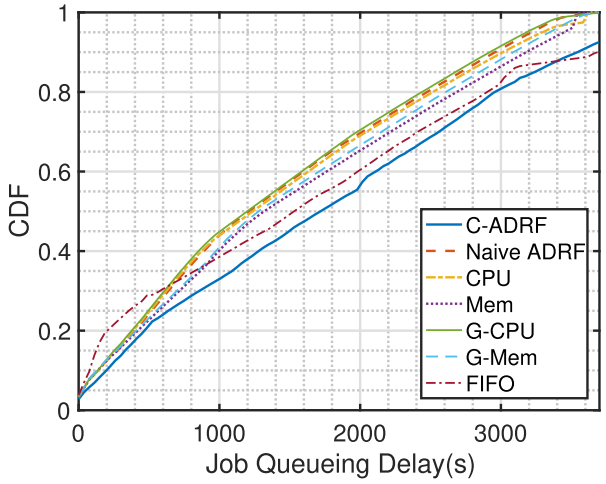


(a) CDF of job queueing delay in Google traces

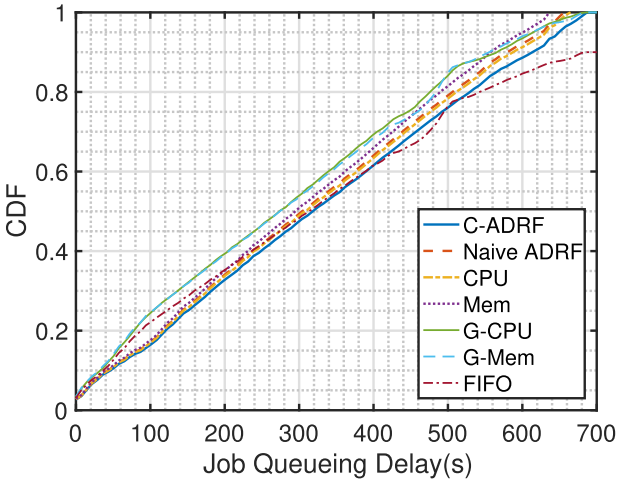


(b) CDF of job queueing delay in Facebook traces

FIGURE 6. CDF of job queueing delay of C-ADRF vs. other 6 algorithms when $\alpha \to 1$.



(a) CDF of job queueing delay in Google traces



(b) CDF of job queueing delay in Facebook traces

FIGURE 7. CDF of job queueing delay of C-ADRF vs. other 6 algorithms when $\alpha = 1.5$.

all the seven online scheduling algorithms in separate traces. As we expected, the job queueing delay of C-ADRF performs worse than the others in most cases. We attribute this to the cumulative nature of C-ADRF while scheduling. C-ADRF has to wait until the cumulative idle resource can support one job from the poorest user. This will lead to a loss on the performance of job queueing delay of C-ADRF. However, this is a slight loss. All results indicate that the near-optimal social welfare performance of C-ADRF is achieved at the expense of only a slight increase in job queueing delay.

VII. CONCLUSION

We proposed ADRF, a multi-resource fair allocation policy that provided near-optimal social welfare. ADRF was a special case of GW-DRF. We proved, both analytically and experimentally, that when $1 < \alpha \leq 2$ in the α-fair utility function, ADRF provided near-optimal social welfare and the upper bound of the gap between the social welfare ADRF provided and the optimal social welfare could be minimized. We further proposed C-ADRF, a non-preemptive job scheduling algorithm to implement ADRF online based on the progressive filling method. Whenever a resource freed up, C-ADRF assigned it to the user i with the lowest current adaptive dominant share. If the idle resources could not satisfy a job of user i, C-ADRF would wait until the cumulative freed up resources could accommodate one job of user i. Through extensive simulation studies, we showed that C-ADRF differed from the optimal solutions in SWM by only 3% using Google traces and 2% using Facebook traces, providing near-optimal social welfare dynamically for non-preemptive datacenter jobs. And this nice social welfare performance was achieved at the expense of only a slight

increase in job queueing delay. For future research directions, we believe C-ADRF is applicable in many other multi-resource contexts for non-preemptive scheduling, such as VM scheduling in hypervisors.

APPENDIX A
PROOF OF LEMMA 6

Proof: Since H_n is a positive semi-definite matrix, according to Cholesky Decomposition, there exists a lower triangular matrix L_n such that $H_n = L_n L_n^T$. Denote l_{ij} as the individual item of L_n on the i-th row and j-th column, where $i, j = 1, 2, \ldots, n$, then we have

$$\begin{cases} l_{ii} = \sqrt{h_{ii} - \sum_{t=1}^{} l_{it}^2}, \\ l_{ij} = \dfrac{h_{ij} - \sum_{t=1}^{j-1} l_{it} l_{jt}}{l_{jj}}, \quad (i > j), \end{cases} \tag{20}$$

where $l_{ii} \geq 0$. If we denote H_n as following

$$H_n = \begin{bmatrix} h_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix}, \tag{21}$$

where H_{12} is a row vector, H_{21} is a column vector and H_{22} is a $(n-1)$-dimensional square matrix. Then another expression of Cholesky Decomposition of H_n can be denoted as following

$$l_{11} = \sqrt{h_{11}} \tag{22a}$$

$$L_{21} = \frac{1}{l_{11}} H_{21} \tag{22b}$$

$$L_{22} L_{22}^T = H_{22} - L_{21} L_{21}^T, \tag{22c}$$

According to (18), if for another matrix H_n' we have $h_{11}' \geq h_{11} \geq 0$, then denote l_{11}' to be equal to $\sqrt{h_{11}'}$, (18) and (22a) together yield

$$l_{11}' = \sqrt{h_{11}'} \geq \sqrt{h_{11}} = l_{11}. \tag{23}$$

At this point, if we let $L_{21}' = \frac{1}{l_{11}'} H_{21}'$, since $l_{11}' \geq l_{11}$ and $0 \geq h_{i1}' \geq h_{i1}$ for $i = 1, 2, \ldots, n$, we have

$$0 \geq L_{21}' = \frac{1}{l_{11}'} H_{21}' \geq \frac{1}{l_{11}} H_{21} = L_{21}. \tag{24}$$

Here, the comparison between two matrices implies the comparison between each pair of elements at the same location of the two matrices. That is, $H' \geq H$ indicates that each item h_{ij}' in H' is larger than or equal to its corresponding element h_{ij} in H (*i.e.*, $h_{ij}' \geq h_{ij}$). Next, we denote

$$L_{22}' L_{22}'^T = H_{22}' - L_{21}' L_{21}'^T. \tag{25}$$

It is easy to verify that $L_{21}' L_{21}'^T$ is a diagonal matrix, and

$$0 \leq L_{21}' L_{21}'^T \leq L_{21} L_{21}^T. \tag{26}$$

Thus, we have

$$H_{22}' - L_{21}' L_{21}'^T \geq H_{22} - L_{21} L_{21}^T. \tag{27}$$

(22c),(25) and (27) together yield

$$L_{22}' L_{22}'^T \geq L_{22} L_{22}^T \tag{28}$$

Therefore, there exists a lower triangular matrix L_n' that can be expressed as following

$$l_{11}' = \sqrt{h_{11}'} \tag{29a}$$

$$L_{21}' = \frac{1}{l_{11}'} H_{21}' \tag{29b}$$

$$L_{22}' L_{22}'^T = H_{22}' - L_{21}' L_{21}'^T, \tag{29c}$$

which is a similar expression of L_n that is shown in Equation (22).

Noticing that $L_{22} L_{22}^T$ is also a Cholesky Decomposition, and the matrix that being decomposed is the $(n-1)$ dimensional matrix of the lower right corner of matrix H_n. Therefore, H_{22} can be further decomposed using Cholesky Decomposition. According to the relationship shown in (18), by repeating the process above, we obtain a lower triangular matrix L_n' in which l_{ii}' can be expressed as following

$$\begin{cases} l_{ii}' = \sqrt{h_{ii}' - \sum_{t=1}^{i-1} l_{it}'^2}, \\ l_{ij}' = \dfrac{h_{ij}' - \sum_{t=1}^{j-1} l_{it}' l_{jt}'}{l_{jj}'}, \quad (i > j), \end{cases} \tag{30}$$

That is, $H_n' = L_n' L_n'^T$. Since all the diagonal elements of L_n' is larger than or equal to 0, it is easy to indicate that $\det(H') = \det(L_n') \det(L_n'^T) \geq 0$. Therefore, H_n' is also a positive semi-definite matrix. ■

APPENDIX B
PROOF OF PROPOSITION 7

Proof: According to the results shown in (16) and (17), our problem becomes to find the optimal solution of β_i in order to minimize the right hand sides of the two formulas accordingly. That is, regardless of the constant terms, we seek to find the optimal β_i with constraints $\beta_i > 0$ such that

$$\min_{\{\beta_i\}} f(\beta_i) = \sum_{i \in \mathcal{A}} \frac{(\beta_i^{\frac{1}{\alpha}} \sum_{j \in \mathcal{A}} d_{jr_{\max}} \beta_j^{-\frac{1}{\alpha}})^{\alpha-1}}{\alpha - 1}. \tag{31}$$

Now we show that the sum objective in (31), which we denote as $f(\beta_i)$ $(i \in \mathcal{A})$ is convex with respect to β_i. We prove the convexity of $f(\beta_i)$ by showing the convexity of each term, $(\beta_i^{\frac{1}{\alpha}} \sum_{j \in \mathcal{A}} d_{jr_{\max}} \beta_j^{-\frac{1}{\alpha}})^{\alpha-1}/(\alpha-1)$, for $i \in \mathcal{A}$.

We denote H_N as the Hessian matrix of each term of $f(\beta_i)$. We will show by induction that H_N is positive semi-definite for all N, which can indicate that (31) is convex. First, for $N = 1$, it is easy to see that the objective of (31) is a constant and hence also convex. Next, for some given $N = n-1$ users, suppose that H_{n-1} is positive semi-definite. Denote by $h_{ii}^{(n-1)}$

its ith diagonal element. Then we have

$$h_{ii}^{(n-1)} = \frac{1}{\alpha^2}[-\beta_i^{-\frac{1}{\alpha}-1}B_{n-1}^{\alpha-1} + (3-\alpha)d_{ir_{\max}}\beta_i^{-\frac{2}{\alpha}-1}B_{n-1}^{\alpha-2} + (\alpha-2)d_{ir_{\max}}^2\beta_i^{-\frac{3}{\alpha}-1}B_{n-1}^{\alpha-3}], \quad (32)$$

where $B_{n-1} = \sum_{i=1}^{n-1} d_{ir_{\max}}\beta_i^{-\frac{1}{\alpha}}$. The positive semi-definiteness of H_{n-1} indicates $h_{ii}^{n-1} \geq 0$. The other element $h_{ij}^{(n-1)}$ in H_{n-1} where $i \neq j$ can be expressed as

$$h_{ij}^{(n-1)} = \frac{1}{\alpha^2}d_{jr_{\max}}\beta_j^{-\frac{1}{\alpha}-1}[(1-\alpha)\beta_i^{-\frac{1}{\alpha}}B_{n-1}^{\alpha-2} + (\alpha-2)d_{ir_{\max}}\beta_i^{-\frac{2}{\alpha}}B_{n-1}^{\alpha-3}]. \quad (33)$$

Now we add one more user to the above $n-1$ users and consider the resultant H_n. We hope that there exits a lower triangular matrix L_n such that $H_n = L_n L_n^T$, where its diagonal element $l_{ii}^{(n)} \geq 0$, $i = 1, 2, \ldots, n$.

We firstly construct a n-dimensional positive semi-definite matrix $\tilde{H}_n$ by adding one column and one row to H_{n-1}, in which the (n,n) entry $\tilde{h}_{nn}^{(n)}$ is a positive number that is smaller than or equal to $h_{nn}^{(n)}$, while the rest newly added entries are zero. That is, in $\tilde{H}_n$ we have

$$\begin{cases} \tilde{h}_{ij}^{(n)} = h_{ij}^{(n-1)}, \quad i,j = 1,\ldots,n-1. \\ \tilde{h}_{nn}^{(n)} \leq h_{nn}^{(n)}. \\ \tilde{h}_{in}^{(n)} = 0, \quad i = 1,\ldots,n. \\ \tilde{h}_{nj}^{(n)} = 0, \quad j = 1,\ldots,n. \end{cases} \quad (34)$$

Thus, H_n' is obviously a positive semi-definite matrix. Then, according to Lemma 6, we know that as long as

$$\begin{cases} h_{ii}^{(n)} \geq \tilde{h}_{ii}^{(n)} \geq 0, \\ \tilde{h}_{ij}^{(n)} \leq h_{ij}^{(n)} \leq 0, \end{cases} \quad (35)$$

holds, H_n is also positive semi-definite. We notice that compared with $h_{ii}^{(n-1)}$ and $h_{ij}^{(n-1)}$, the only element that would be changed in $h_{ii}^{(n)}$ and $h_{ij}^{(n)}$ is B_n (*i.e.*, B_n replaces B_{n-1}). Since $d_{ir_{\max}}$ and β_i are both positive, we have

$$B_{n-1} < B_n \quad (36)$$

Therefore, when $h_{ii}^{(n)} \geq \tilde{h}_{ii}^{(n)}$, as we already have $h_{nn}^{(n)} \geq \tilde{h}_{nn}^{(n)}$, what we need to prove is $h_{ii}^{(n)} \geq h_{ii}^{(n-1)} = \tilde{h}_{ii}^{(n)}$ ($i = 1, \ldots, n-1$). Therefore, we have

$$\begin{aligned} &\frac{1}{\alpha^2}[-\beta_i^{-\frac{1}{\alpha}-1}B_{n-1}^{\alpha-1} + (3-\alpha)d_{ir_{\max}}\beta_i^{-\frac{2}{\alpha}-1}B_{n-1}^{\alpha-2} \\ &\quad + (\alpha-2)d_{ir_{\max}}^2\beta_i^{-\frac{3}{\alpha}-1}B_{n-1}^{\alpha-3}] \\ &\geq \frac{1}{\alpha^2}[-\beta_i^{-\frac{1}{\alpha}-1}B_n^{\alpha-1} + (3-\alpha)d_{ir_{\max}}\beta_i^{-\frac{2}{\alpha}-1}B_n^{\alpha-2} \\ &\quad + (\alpha-2)d_{ir_{\max}}^2\beta_i^{-\frac{3}{\alpha}-1}B_n^{\alpha-3}], \end{aligned} \quad (37)$$

After eliminating $-\frac{1}{\alpha^2}\beta_i^{-\frac{3}{\alpha}-1}d_{ir_{\max}}$ on the both sides of (37), we have

$$B_n^{\alpha-3}(T_n+\alpha-2)(T_n-1) \leq B_{n-1}^{\alpha-3}(T_{n-1}+\alpha-2)(T_{n-1}-1), \quad (38)$$

where

$$\begin{aligned} T_n &= d_{ir_{\max}}^{-1}\beta_i^{\frac{1}{\alpha}}B_n \\ &= d_{ir_{\max}}^{-1}\beta_i^{\frac{1}{\alpha}}\sum_{i=1}^{n} d_{ir_{\max}}\beta_i^{-\frac{1}{\alpha}} \\ &= 1 + \sum_{j\in\mathcal{A}, j\neq i} d_{jr_{\max}}\beta_j^{-\frac{1}{\alpha}} \\ &> 1. \end{aligned} \quad (39)$$

Furthermore, since $B_n \geq B_{n-1}$ (see (36)), we have $T_n \geq T_{n-1}$. Therefore, if $\alpha > 1$ we always have

$$\begin{cases} T_n + \alpha - 2 \geq T_{n-1} + \alpha - 2 \geq 0, \\ T_n - 1 \geq T_{n-1} - 1 \geq 0. \end{cases} \quad (40)$$

This indicates

$$(T_n+\alpha-2)(T_n-1) \geq (T_{n-1}+\alpha-2)(T_{n-1}-1) \quad (41)$$

Combine (41) with (38), we have

$$B_n^{\alpha-3} \leq B_{n-1}^{\alpha-3}. \quad (42)$$

Since $B_n > B_{n-1}$, we indicate that

$$\alpha - 3 \leq 0. \quad (43)$$

To sum up, when $1 < \alpha \leq 3$, $h_{ii}^{(n)} \geq h_{ii}^{(n-1)}$ can always be satisfied.

Next, when $\tilde{h}_{ij}^{(n)} \leq h_{ij}^{(n)} \leq 0$, as we already have $\tilde{h}_{in}^{(n)} = 0$ ($i = 1, \ldots, n$) and $\tilde{h}_{nj}^{(n)} = 0$ ($j = 1, \ldots, n$), what we need to prove is that $h_{ij}^{(n-1)} = \tilde{h}_{ij}^{(n)} \leq h_{ij}^{(n)}$ ($i, j = 1, \ldots, n-1$ and $i \neq j$). According to (33), it is obvious to notice that if

$$1 < \alpha \leq 2, \quad (44)$$

we will always have

$$\begin{cases} 1 - \alpha < 0 \\ \alpha - 2 \leq 0. \end{cases} \quad (45)$$

and

$$\begin{cases} B_{n-1}^{\alpha-2} \geq B_n^{\alpha-2} \\ B_{n-1}^{\alpha-3} \geq B_n^{\alpha-3} \end{cases} \quad (46)$$

According to the expression of $h_{ij}^{(n-1)}$ shown in (33), (35) guarantees that $h_{ij}^{(n-1)}$ and $h_{ij}^{(n-1)}$ are always negative. Meanwhile, (46) guarantees that $h_{ij}^{(n-1)} \leq h_{ij}^{(n)}$. Therefore, when $1 < \alpha \leq 2$, we always have $h_{ij}^{(n-1)} \leq h_{ij}^{(n)} \leq 0$.

To sum up, (43) and (44) together yield

$$1 < \alpha \leq 2 \quad (47)$$

At this point, according to Lemma 6, H_n is also a positive semi-definite matrix. Therefore, when $1 < \alpha \leq 2$, each term of the objective function in (31) is convex. Thus the objective function in (31) is convex.

Now we are able to compute the stationary points of the objective function in (31). When $1 < \alpha \leq 2$, the first derivative of the objective function in (31) is

$$\begin{aligned}\frac{\partial f(\beta_i)}{\partial \beta_i} &= (\beta_1^{\frac{1}{\alpha}} B_N)^{\alpha-2}[-\frac{1}{\alpha}\beta_1^{\frac{1}{\alpha}} d_{ir_{\max}} \beta_i^{-\frac{1}{\alpha}-1}] \\ &\quad + \ldots + \frac{1}{\alpha}(\beta_i^{\frac{1}{\alpha}} B_N)^{\alpha-2}[\beta_i^{\frac{1}{\alpha}-1} B_N - d_{ir_{\max}} \beta_i^{-1}] \\ &\quad + \ldots + (\beta_N^{\frac{1}{\alpha}} B_N)^{\alpha-2}[-\frac{1}{\alpha}\beta_N^{\frac{1}{\alpha}} d_{ir_{\max}} \beta_i^{-\frac{1}{\alpha}-1}] \quad (48a) \\ &= B_N^{\alpha-2}(\beta_i^{-\frac{1}{\alpha}} B_N - d_{ir_{\max}} \beta_i^{-1-\frac{1}{\alpha}} \sum_{j\in\mathcal{A}} \beta_j^{1-\frac{1}{\alpha}}). \quad (48b)\end{aligned}$$

Let $\frac{\partial f(\beta_i)}{\partial \beta_i}$ to be zero, then we have the stationary point of $f(\beta_i)$ is $\beta_i = d_{ir_{\max}}$. This is because: (1) the first term of (48b) is always larger than zero; (2) when $\beta_i = d_{ir_{\max}}$, we have $B_N = \sum_{j\in\mathcal{A}} \beta_j^{1-\frac{1}{\alpha}}$ and $\beta_i^{-\frac{1}{\alpha}} = d_{ir_{\max}} \beta_i^{-1-\frac{1}{\alpha}}$, thus the second term of (48b) is equal to zero.

According to the concavity of the objective functions in the two problems, $\beta_i = d_{ir_{\max}}$ is also the optimal point that minimize the value of the upper bound of the GAP concluded in Proposition 7. ■

REFERENCES

[1] I. Menache, A. Ozdaglar, and N. Shimkin, "Socially optimal pricing of cloud computing resources," in *Proc. 5th Int. ICST Conf. Perform. Eval. Methodol. Tools*, 2011, pp. 322–331.

[2] L. Zhang, Z. Li, and C. Wu, "Dynamic resource provisioning in cloud computing: A randomized auction approach," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 433–441.

[3] H. Li, C. Wu, Z. Li, and F. C. M. Lau, "Virtual machine trading in a federation of clouds: Individual profit and social welfare maximization," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1827–1840, Jun. 2016.

[4] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. M. Lau, "Online auctions in IaaS clouds: Welfare and profit maximization with server costs," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 1034–1047, Apr. 2017.

[5] W. Shi, C. Wu, and Z. Li, "An online auction mechanism for dynamic virtual cluster provisioning in geo-distributed clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 677–688, Mar. 2017.

[6] J. Li, Y. Zhu, J. Yu, C. Long, G. Xue, and S. Qian, "Online auction for IaaS clouds: Towards elastic user demands and weighted heterogeneous VMs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 9, pp. 2075–2089, Sep. 2018.

[7] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in *Proc. Eur. Conf. Comput. Syst. (EuroSys)*, 2010, pp. 265–278.

[8] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in *Proc. USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2011, p. 22.

[9] A. Ghodsi, M. Zaharia, S. Shenker, and I. Stoica, "Choosy: Max-min fair sharing for datacenter jobs with constraints," in *Proc. 8th ACM Eur. Conf. Comput. Syst. EuroSys*, 2013, pp. 365–378.

[10] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[11] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization," *IEEE Trans. Cloud Comput.*, vol. 1, no. 2, pp. 158–171, Jul./Dec. 2013.

[12] T. Hu, Y. Zhang, and M. Cheng, "Pricing strategy of car-hailing platform with maximizing social welfare," in *Proc. 6th Int. Conf. Frontiers Ind. Eng. (ICFIE)*, Sep. 2019, pp. 33–39.

[13] N. Chen and G. Gallego, "Welfare analysis of dynamic pricing," *Manage. Sci.*, vol. 65, no. 1, pp. 139–151, Jan. 2019.

[14] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, and X. Wang, "How to bid the cloud," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 71–84, Sep. 2015.

[15] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in *Proc. USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, vol. 11, 2011, p. 24.

[16] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data Networks*, vol. 2. Upper Saddle River, NJ, USA: Prentice-Hall, 1992.

[17] A. N. Toosi, K. Vanmechelen, K. Ramamohanarao, and R. Buyya, "Revenue maximization with optimal capacity control in infrastructure as a service cloud markets," *IEEE Trans. Cloud Comput.*, vol. 3, no. 3, pp. 261–274, Jul. 2015.

[18] M. Hadji and D. Zeghlache, "Mathematical programming approach for revenue maximization in cloud federations," *IEEE Trans. Cloud Comput.*, vol. 5, no. 1, pp. 99–111, Jan. 2017.

[19] M. Wardat, M. Al-Ayyoub, Y. Jararweh, and A. A. Khreishah, "Cloud data centers revenue maximization using server consolidation: Modeling and evaluation," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 172–177.

[20] W. You, L. Jiao, J. Li, and R. Zhou, "Scheduling DDOS cloud scrubbing in ISP networks via randomized online auctions," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2020.

[21] X. Yi, F. Liu, Z. Li, and H. Jin, "Flexible instance: Meeting deadlines of delay tolerant jobs in the cloud with dynamic pricing," in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2016, pp. 415–424.

[22] H. Zhang, H. Jiang, B. Li, F. Liu, A. V. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 805–818, Mar. 2016.

[23] Y. Jiao, P. Wang, D. Niyato, and K. Suankaewmanee, "Auction mechanisms in cloud/fog computing resource allocation for public blockchain networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 9, pp. 1975–1989, Sep. 2019.

[24] Q. Li, C. Huang, H. Bao, B. Fu, and X. Jia, "A game-based combinatorial double auction model for cloud resource allocation," in *Proc. 28th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2019, pp. 1–8.

[25] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[26] G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in *Proc. Int. Conf. Manage. Data SIGMOD*, 2010, pp. 135–146.

[27] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed data-parallel programs from sequential building blocks," in *Proc. ACM Symp. Oper. Syst. Princ. (SIGOPS)*, 2007, vol. 41, no. 3, pp. 59–72.

[28] V. K. Vavilapalli, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, E. Baldeschwieler, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, and H. Shah, "Apache Hadoop YARN: Yet another resource negotiator," in *Proc. 4th Annu. Symp. Cloud Comput. (SOCC)*, 2013, p. 5.

[29] R. Srikant and L. Ying, *Communication Networks: An Optimization, Control, and Stochastic Networks Perspective*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[30] D. C. Parkes, A. D. Procaccia, and N. Shah, "Beyond dominant resource fairness: Extensions, limitations, and indivisibilities," *ACM Trans. Econ. Comput.*, vol. 3, no. 1, pp. 1–22, 2015.

[31] W. Wang, B. Li, and B. Liang, "Dominant resource fairness in cloud computing systems with heterogeneous servers," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 583–591.

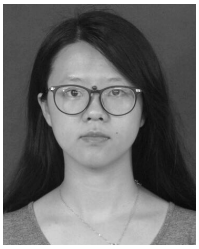
[32] A. A. Bhattacharya, D. Culler, E. Friedman, A. Ghodsi, S. Shenker, and I. Stoica, "Hierarchical scheduling for diverse datacenter workloads," in *Proc. 4th Annu. Symp. Cloud Comput. (SOCC)*, 2013, pp. 1–15.

[33] W. Wang, B. Li, B. Liang, and J. Li, "Multi-resource fair sharing for datacenter jobs with placement constraints," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal. (SC)*, Nov. 2016, p. 86.

[34] P. Marbach, "Priority service and max-min fairness," in *Proc. 21st Annu. Joint Conf. IEEE Comput. Commun. Societies*, vol. 1, Jun. 2002, pp. 266–275.

[35] 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.

[36] G. Zhang, R. Lu, and W. Wu, "Multi-resource fair allocation for cloud federation," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun., IEEE 17th Int. Conf. Smart City, IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Aug. 2019, pp. 2189–2194.
[37] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2012, pp. 15–28.
[38] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: Fair scheduling for distributed computing clusters," in *Proc. ACM SIGOPS 22nd Symp. Oper. Syst. Princ. (SOSP)*, 2009, pp. 261–276.
[39] W. Wang, B. Liang, and B. Li, "Low complexity multi-resource fair queueing with bounded delay," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 1914–1922.
[40] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format+ schema," Google Inc., Menlo Park, CA, USA, White Paper, 2011, pp. 1–14.
[41] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*, vol. 116. Reading, MA, USA: Addison-Wesley, 1997.



CHAOQUN YOU (Member, IEEE) received the B.S. degree in communication engineering from the University of Electronic Science and Technology of China (UESTC), in July 2013, where she is currently pursuing the Ph.D. degree. From October 2015 to September 2017, she was an Academic Guest with the Department of Electronic Computer Engineering, University of Toronto. Her research interests include data center networks (DCN), network function virtualization (NFV), and distributed machine learning (DML).

CHENG REN received the B.S., M.S., and Ph.D. degrees from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2006. She is currently a Lecturer with Southwest Petroleum University. Her research interests include software-defined networking and network resource allocation.

LEMIN LI received the B.S. degree in electrical engineering from Shanghai Jiao Tong University (SJTU), in 1952. Then, he was with the Department of Electrical Communications, SJTU, until 1956. Since 1956, he has been with the Chengdu Institute of Radio Engineering (now UESTC). From August 1980 to August 1982, he was a Visiting Scholar with The University of California at San Diego (UCSD). His current research interest includes communication networks.