Fair Multi-resource Allocation in Heterogeneous Servers with an External Resource Type

Erfan Meskar, *Student Member, IEEE,* and Ben Liang, *Fellow, IEEE*

Abstract—This paper considers the problem of fair allocation of multiple types of resources in heterogeneous servers, along with a resource type external to those servers. Our work is motivated by the need for fair multi-resource allocation in mobile edge computing (MEC), where the users must upload their tasks over a single dedicated wireless communication link that exists outside the computing servers. We propose a fair multi-resource allocation mechanism for this environment, termed Task Share Fairness with External Resource (TSF-ER), which finds the Kalai-Smorodinsky bargaining solution satisfying important fairness properties. We show that TSF-ER is envy-free, Pareto optimal, and strategy-proof, and it satisfies the property of sharing incentive. Large-scale simulation driven by Google and Alibaba cluster trace further shows that TSF-ER significantly outperforms the existing utilitarian, Nash social welfare maximizer, and egalitarian solutions, leading to fairer resource allocation while maintaining a high level of resource utilization.

Index Terms—Fair resource allocation, multiple resource types, external resource, heterogeneous computing servers.

1 Introduction

In the long history of mathematically rigorous fair division, there have been many works focusing on the division of a set of items among a set of agents. The goal is to find a division of the items that is fair to all agents. This problem has been considered from different perspectives in economics and computer science, and arises in various real-world settings: auctions, airport traffic management [1], and computing resource allocation in cloud servers and routers [2], [3], [4]. Developing a fair division mechanism is of immense significance to guarantee quality of experience for different agents. Under-allocation degrades an agent's quality of experience, and over-allocation would adversely impact other agents.

In systems with a *single* resource, one of the most popular allocation policy is max-min fairness. Max-min fairness is an egalitarian approach by which the resource allocated to an agent can be increased only if it will not decrease the allocation of an agent with already smaller allocation. However, egalitarianism may conflict with the fairness and efficiency properties in systems with multiple resources.

To evaluate an allocation policy in a multi-resource environment, we observe whether it satisfies several core properties of a fair and efficient resource allocation policy [2], [3], [4], [5], [6]:

- Envy-Freeness (EF): No agent prefers the allocation of another agent.
- Pareto Optimality (PO): It should be impossible to increase the resource amount of an agent without decreasing the allocation of another agent.
- Strategy-Proofness (SP): An agent cannot improve its allocation by lying, which provides incentive compatibility.
- Sharing Incentive (SI): No agent prefers to split the total resources equally.

- *Erfan Meskar and Ben Liang are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada.*
E-mail: {emeskar, liang}@ece.utoronto.ca

This work was supported in part by a grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

EF and PO as fairness and efficiency concepts were introduced by Foley [7] and Varian [8] and have been studied extensively since then [9], [10]. Unlike egalitarian social welfare, EF does not require interpersonal comparability of individual preferences. With SP, we ensure the agents' truthfulness. Truthfulness is a requirement especially in environments without monetary transfers [11]. SI, also known as proportionality, which motivates agents' participation, requires that the agents prefer their allocation to their proportional share of the resources. If SI is not satisfied by an allocation rule, the agents have no incentive to cooperate and prefer to split the resources equally to receive their guaranteed shares. SI dates back to at least the work of Steinhaus [12] in the context of cake-cutting, and together with maximin share guarantee, which is the natural relaxation of SI in indivisible items environments, have been studied extensively since then [13] - [21].

It may be impossible to satisfy all properties in some cases. For instance, in the 2-agent case with continuous, increasing, and strictly quasi-concave utilities, any Pareto optimal and strategy-proof allocation policy violates EF and SI [22]. An interesting question is whether it is possible to satisfy all properties for more restricted types of utility functions. Cobb-Douglas utilities and Leontief utilities are among the appealing restricted domains in economics and computer science [2], [6], [23], [24].

With Cobb-Douglas utilities, Hashimoto proved that even in the 2-agent case, any Pareto optimal and strategy-proof allocation policy violates EF and SI [25]. Zahedi *et al.* relaxed the notion of SP and proved that, for Cobb-Douglas utilities, the Nash bargaining solution (*i.e.*, maximizing the product of utilities [26], [27]) satisfies PO, EF, and SI, and the agents have no incentive to lie about their utility function when there are many agents in the system [23], [28]. Nicoló, on the other hand, presented resource conservative mechanisms (*i.e.*, mechanisms that allocate resources entirely) in 2-agent, 2-resource systems with Leontief utilities that can satisfy the four properties altogether [29]. Furthermore, stronger results are obtainable when only non-wasteful allocation policies are considered [30].

Ghodsi *et al.* studied the problem of fair multi-resource allocation in a cloud computing server with l resources and n agents with Leontief utilities. They proposed Dominant Resource Fairness (DRF), a non-wasteful resource allocation policy, which can satisfy the four properties altogether [2]. DRF computes the share of demanded resources for each agent and finds each agent's dominant share and the resource corresponding to the dominant share. It then applies max-min fairness across agents' dominant shares. DRF can be interpreted as a Kalai-Smorodinsky (KS) bargaining solution [31], [32]. It finds the lexicographic maxmin solution after a certain normalization of utilities [6]. Hence, Ghodsi *et al.* [2] showed that the KS solution satisfies EF, PO, SP, and SI when the agents have Leontief utilities. Note that the analysis in [2] assumes that the agents have non-zero demands for all the resources. Parkes *et al.* [5] generalized the results in [2] to the case of agents with Leontief utilities and possibly zero demands for some resources by proposing a multi-round DRF allocation.

Wang *et al.* attempted to satisfy the four properties in cloud computing systems with heterogeneous servers [3], [33]. Wang *et al.* proposed DRFH in [3] and defined the dominant resource of an agent based on the aggregate of all the resources. DRFH computes the max-min optimal allocation regarding the dominant global shares, which is the allocation share of the dominant resource. They showed that DRFH satisfies EF, PO, and SP, but fails to satisfy SI. They further proposed Task Sharing Fairness (TSF) in [33], which extends the KS solution to cloud computing systems with heterogeneous servers, with additional accommodations for task placement constraints. They proved that TSF satisfies all four properties.

In this paper, we study the problem of fair resource allocation in systems with heterogeneous servers and an external resources to access the servers. A motivating example is Mobile Edge Computing (MEC) systems with heterogeneous servers. The computing tasks in MEC require multiple types of resources in the computing servers (*e.g.*, memory and CPU cores) [34]. A MEC system is possibly constructed from a variety of server classes. These servers may have different processing capabilities, memory sizes, and storage spaces. Moreover, hardware upgrades, *i.e.*, adding new servers and phasing out existing ones, increase the server heterogeneity. In addition to these computing resources, the task input data and execution results are sent through a shared wireless communication link to and from the MEC servers. The wireless communication link is a dedicated resource that exists outside of the computing servers and shared by all agents to access the servers (*i.e.*, external resource).

Developing a fair resource allocation mechanism is of immense significance in complex multi-resource computing systems such as MEC. Different computing tasks can consume vastly various amounts of different resources. For instance, video analysis, language translation, face recognition, and augmented reality applications typically have CPU-intensive tasks, graph analytics and data indexing may have memory-bound tasks, and vehicle-to-infrastructure communication services can bottleneck on wireless communication link bandwidth [35], [36]. Server heterogeneity and diversity across the resource demands present challenges to developing a fair resource allocation mechanism. Although DRFH and TSF consider resource allocation across heterogeneous servers, they cannot be directly applied here. These mechanisms require a server where every type of resource is contained. However, in our new environment, there is a single dedicated external resource (*e.g.*, wireless communication link) that exists outside of the computing servers.

The main contribution of this paper are as follows:

- We show that, in the aforementioned computing environment with an external resource, no existing multi-resource allocation rules can simultaneously satisfy the fairness and efficiency properties of EF, PO, SP, and SI. In particular, there is no direct extension to DRFH or TSF to satisfy PO or SI, no matter how the external resource is split among the servers.
- We propose Task Share Fairness with External Resource (TSF-ER) which applies max-min fairness across users' normalized number of allocated tasks, without explicitly splitting the external resource among the servers. we show that it satisfies all of EF, PO, SP, and SI.
- We further consider practical online implementation of TSF-ER with invisible tasks and test its performance and efficiency via trace-driven simulation. The simulation results show that TSF-ER attains more important fairness properties than all existing solutions, without degrading resource utilization.

The organization of this paper is as follows. In Sec. 2 we describe the system model and allocation properties. In Sec. 3, we analyze naive extension of the existing allocation rules, DRFH, TSF, and DRF-ER [37]. We show that none of these solutions can achieve all of EF, PO, SP, and SI. In Sec. 4, we present the proposed TSF-ER fair allocation policy. We prove that TSF-ER satisfies all of the desirable properties. We evaluate the performance of TSF-ER via trace-driven simulations in Sec. 6. In Sec. 7, we summarize the existing multi-resource fair allocation solutions followed by conclusion in Sec. 8.

2 System Model and Allocation Properties

We consider a set of heterogeneous computing servers that are accessible over a shared external resource (*e.g.*, communication channel). We denote the set of computing servers by $\mathcal{S}$ and the set of resources in the servers (*e.g.*, CPU and memory) by $\mathcal{R}$. The capacity of server s for resource r is denoted by $c_{s,r}$, and its capacity profile is $\mathbf{c}_s = (c_{s,r})_{r\in\mathcal{R}}$. We denote the *computational* capacity profile of this environment by $\mathbf{c} = (\mathbf{c}_s)_{s\in\mathcal{S}}$.

We assume that all users share an additional external resource, usually a communication link, to access the servers. Hence, the external resource is a single dedicated resource that exists outside of the computing servers. In the example of MEC, illustrated in Figure 2, the wireless communication link is a single resource that all users share when they upload their tasks to the servers. We denote the system capacity for this external resource, *e.g.*, the total link bandwidth in MEC, by c^{er}. Let $\hat{\mathcal{R}}$ be the augmented set of resources, which is constructed by adding the external resource to the set of computational resources $\mathcal{R}$.

Let $\mathcal{J}$ denote the set of users in the system. They submit their tasks upon arrival in the system, and the number of tasks submitted by user j is denoted by n_j. For any resource $r \in \hat{\mathcal{R}}$, user j requires $d_{j,r}$ units of the resource r per task. For instance, consider user j that requires 2 CPU cores, 1 GB of memory, and 150 MHz of bandwidth per task. With an allocation of 6 CPU cores and 4 GB of memory on a server and 300 MHz of bandwidth, it can execute $\min\{6/2, 4/1, 300/150\} = 2$ tasks. Without loss of generality, assume that the aggregate capacity

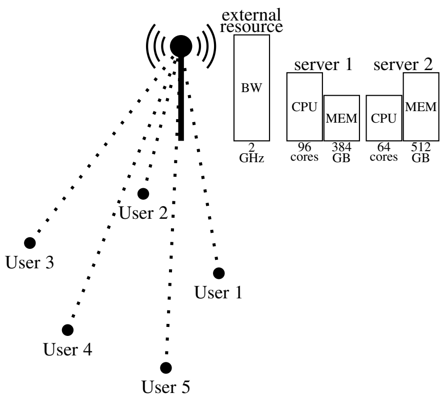


Fig. 1. Illustration of system model with MEC as example.

of any computational resource $r \in \mathcal{R}$ is normalized to one (*i.e.*, $\sum_{s\in\mathcal{S}} c_{s,r} = 1$), so is the capacity of the external resource (*i.e.*, $c^{\text{er}} = 1$).[1] Hence, $d_{j,r}$ can be interpreted as the share of the aggregate capacity of resource r that user j requires per task.[2] Let $\mathbf{d}_j = (d_{j,r})_{r\in\hat{\mathcal{R}}}$ denote the demand profile of user j.

We denote the share of external resource that is allocated to user j by A_j^{er}, and the share of computational resource r that is allocated to user j in server s by $A_{j,s,r}$. If we were not concerned about the external resource (*e.g.*, consider a parallel computing environment in which there exists no external resource), the number of tasks that user j could execute on server s would have been $\min_{r\in\mathcal{R}} \left\{\frac{A_{j,s,r}}{d_{j,r}}\right\}$ and its total number of executable task would have been $\sum_{s\in\mathcal{S}} \min_{r\in\mathcal{R}} \left\{\frac{A_{j,s,r}}{d_{j,r}}\right\}$. However, user j's demand on the external resource bounds the total number of executable tasks by $\frac{A_j^{\text{er}}}{d_{j,\text{er}}}$. Moreover, the number of tasks that a user can execute should not exceed the number of submitted tasks by that user. Thus, given the allocated share of the computational resources and the external resource, the number of tasks that user j can execute is $\min\left\{\frac{A_j^{\text{er}}}{d_{j,\text{er}}}, \sum_{s\in\mathcal{S}} \min_{r\in\mathcal{R}} \left\{\frac{A_{j,s,r}}{d_{j,r}}\right\}, n_j\right\}$.

We denote the utility function of user j by u_j, which is a strictly increasing function of the number of tasks that user j can execute. Hence,

$$u_j\left(\boldsymbol{A}_j\right) = U_j\left(\min\left\{\frac{A_j^{\text{er}}}{d_{j,\text{er}}}, \sum_{s\in\mathcal{S}} \min_{r\in\mathcal{R}}\left\{\frac{A_{j,s,r}}{d_{j,r}}\right\}, n_j\right\}\right),$$

where $U_j(.)$ is a univariate strictly increasing function and $\boldsymbol{A}_j = \left(\left((A_{j,s,r})_{r\in\mathcal{R}}\right)_{s\in\mathcal{S}}, A_j^{\text{er}}\right)$. This form of the utility function suggests that user j prefers allocation $\boldsymbol{A}_j$ to $\boldsymbol{A}'_j$ if and only if it can execute more tasks with $\boldsymbol{A}_j$. Since all of the properties in this paper are merely based on ordinal preferences of the users over their allocations, without loss of generality, we can equivalently redefine the utility function as the number of tasks a user can execute, *i.e.*,

$$u_j\left(\boldsymbol{A}_j\right) = \min\left\{\frac{A_j^{\text{er}}}{d_{j,\text{er}}}, \sum_{s\in\mathcal{S}} \min_{r\in\mathcal{R}}\left\{\frac{A_{j,s,r}}{d_{j,r}}\right\}, n_j\right\}. \quad (1)$$

1. We can achieve this by scaling the resource capacity and users demand.
2. The share of the aggregate capacity of resource r is $\frac{d_{j,r}}{\sum_{s\in\mathcal{S}} c_{s,r}}$, which is equal to $d_{j,r}$ when the aggregate capacity of computational resource $r \in \mathcal{R}$ is set to one. The same argument applies to the external resource demand.

Let $\mathbf{u} = (u_j)_{j\in\mathcal{J}}$ be the utility profile. An environment E is defined as follows.

Definition 1 (**Environment**). *An environment E is a tuple $\left(\mathcal{J}, \mathcal{S}, \hat{\mathcal{R}}, \mathbf{u}, \mathbf{c}, c^{er}\right)$. Although c^{er} is set to 1, we specify it in the definition of the environment to emphasize that there exists an external resource.*

An allocation rule is a mechanism that assigns resource to the users for each environment E. Note that the resource allocation must not exceed the system capacity. Let $x_{j,s}$ denote the number of tasks allocated to user j on server s and $\mathbf{x} \in \mathbb{R}_+^{|\mathcal{J}|\times|\mathcal{S}|}$ denote the task allocation matrix. Note that the users may receive tasks from more than one server. The allocation $(\boldsymbol{A}_j)_{j\in\mathcal{J}}$ is non-wasteful if there exists no unused resource, *i.e.*, $A_{j,s,r} = x_{j,s}\, d_{j,r}$ for any $s \in \mathcal{S}$ and $r \in \mathcal{R}$ and $A_j^{\text{er}} = \sum_{s\in\mathcal{S}} x_{j,s}\, d_{j,\text{er}}$. We restrict ourselves to non-wasteful allocations. Thus, allocating resource and task are equivalent in our model. Given E, the set of feasible task allocations for this environment is defined as

$$\begin{aligned}\mathcal{X}(E) = \Big\{\mathbf{x} \in \mathbb{R}_+^{|\mathcal{J}|\times|\mathcal{S}|} \mid & \sum_{u\in\mathcal{J}}\sum_{s\in\mathcal{S}} x_{u,s}\, d_{u,\text{er}} \le 1,\\ & \sum_{u\in\mathcal{J}} x_{u,s}\, d_{u,r} \le c_{s,r}, \forall r \in \mathcal{R},\, s \in \mathcal{S},\\ & \sum_{s\in\mathcal{S}} x_{j,s} \le n_j, \forall j \in \mathcal{J}\Big\}. \end{aligned} \quad (2)$$

Definition 2 (**Non-wasteful Allocation Rule**). *A non-wasteful allocation rule λ assigns to each environment E a non-wasteful feasible allocation. We denote the number of tasks allocated to user j in server s by $\lambda_{j,s}(E)$. We denote the share of computational resource r that is allocated to user j in server s by $\Lambda_{j,s,r}(E)$, and the share of external resource that is allocated to user j by $\Lambda_j^{er}(E)$. Since λ is a non-wasteful allocation, we have*

$$\Lambda_{j,s,r}(E) = \lambda_{j,s}(E)\, d_{j,r}, \quad (3)$$
$$\Lambda_j^{er}(E) = \sum_{s\in\mathcal{S}} \lambda_{j,s}(E)\, d_{j,er}, \quad (4)$$
$$\sum_{s\in\mathcal{S}} \lambda_{j,s}(E) \le n_j. \quad (5)$$

The resource allocation profile of user j derived by rule λ is denoted by $\boldsymbol{\Lambda}_j(E) = \left(\left((\Lambda_{j,s,r}(E))_{r\in\mathcal{R}}\right)_{s\in\mathcal{S}}, \Lambda_j^{er}(E)\right)$.

Following the terminology of DRF [2], we define the following essential fairness-related properties: envy-freeness, strategy proofness, and sharing incentive. At the same time, the proposed scheme should satisfy Pareto optimality to ensure efficient resource utilization.

Definition 3 (**Envy-Freeness (EF)**). *An allocation rule λ satisfies EF when no user prefers the allocation of another user,* i.e.,

$$u_j\left(\boldsymbol{\Lambda}_k(E)\right) \le u_j\left(\boldsymbol{\Lambda}_j(E)\right),$$

for any environment $E = \left(\mathcal{J}, \mathcal{S}, \hat{\mathcal{R}}, \mathbf{u}, \mathbf{c}, c^{er}\right)$ and $j, k \in \mathcal{J}$.

Definition 4 (**Pareto Optimality (PO)**). *Non-wasteful allocation rule λ is Pareto optimal, if for any environment E, there exists no feasible task allocation $\mathbf{y} \in \mathcal{X}(E)$ such that $\sum_{s\in\mathcal{S}} y_{i,s} \ge u_i(\boldsymbol{\Lambda}_i)$ for all $i \in \mathcal{J}$, and $\sum_{s\in\mathcal{S}} y_{j,s} > u_j(\boldsymbol{\Lambda}_j)$ for some $j \in \mathcal{J}$.*

Definition 5 (**Strategy-Proofness (SP)**). *Non-wasteful allocation rule λ satisfies SP if no user benefits from reporting fake demand. Let $E = \left(\mathcal{J}, \mathcal{S}, \hat{\mathcal{R}}, \mathbf{u}, \mathbf{c}, c^{er}\right)$ be an arbitrary environment in which user j reports its true demand, i.e., $d_{j,r}$, for all resources in $\hat{\mathcal{R}}$. Moreover, let $E' = \left(\mathcal{J}, \mathcal{S}, \hat{\mathcal{R}}, \mathbf{u}', \mathbf{c}, c^{er}\right)$ be the environment in which user j reports a fake demand, i.e., $d'_{j,r}$, where $d'_{j,r} \neq d_{j,r}$ for some resource in $\hat{\mathcal{R}}$. Non-wasteful allocation rule λ satisfies SP if*

$$u_j\left(\mathbf{\Lambda}_j\left(E\right)\right) \geq u_j\left(\mathbf{\Lambda}_j\left(E'\right)\right),$$

for any user and any environment E and E' as described above.

In a single server environment, a non-wasteful allocation rule satisfies sharing incentive if each user receives at least as many tasks as when resources are equally shared among users [2]. Analogously, we define sharing incentive in multi-server environments by allocating each resource on each server equally among the users.

Definition 6 (**Sharing Incentive (SI)**). *Non-wasteful allocation rule λ satisfies SI, if for any environment E, the total number of task each user receives by λ is at least as much as the total number of task each user receives in the equal share, i.e.,*

$$u_j\left(\mathbf{\Lambda}_j\left(E\right)\right) \geq \mathit{min}\left\{\frac{1/|\mathcal{J}|}{d_{j,er}}, \sum_{s\in\mathcal{S}} \underset{r\in\mathcal{R}}{\mathit{min}}\left\{\frac{c_{s,r}/|\mathcal{J}|}{d_{j,r}}\right\}, n_j\right\}.$$

We note that EF defines inter-user fairness, PO measures resource utilization, SP ensures user truthfulness, and SI motivates user participation. Our objective is to develop a multi-resource fair allocation scheme that retains all four aforementioned properties. Table 1 summarizes important notations used in the paper.

3 Existing Allocation Rules

In this section, we study the three well-known resource allocation rules, *i.e.*, utilitarian, Nash social welfare maximizer, and egalitarian, and their extensions to our environment. We will show that none of them simultaneously satisfies the EF, PO, SP, and SI properties.

3.1 Utilitarian Approach

For any environment E, the utilitarian approach finds the allocation that maximizes the sum of utilities, *i.e.*,

$$\text{arg max}\left\{\sum_{j\in\mathcal{J}}\sum_{s\in\mathcal{S}} x_{j,s} \mid \mathbf{x} \in \mathcal{X}(E)\right\}.$$

While utilitarianism has several philosophically appealing axiomatic characterizations, it implicitly assumes that the utility functions are interpersonally comparable. Hence, the users can easily manipulate the utilitarian allocation rule by applying affine transformations to their utility functions. One way to circumvent this issue is by normalizing the utility functions to range from zero to one, then applying the utilitarian allocation rule to the normalized utility functions. This yields relative utilitarianism [38] - [43].

Formally, let η_j be the maximum number of tasks that user j can execute by monopolizing the resources, *i.e.*,

$$\eta_j = \min\left\{\sum_{s\in\mathcal{S}} \underset{r\in\mathcal{R}}{\min}\left\{\frac{c_{s,r}}{d_{j,r}}\right\}, \frac{1}{d_{j,\text{er}}}\right\}. \quad (6)$$

TABLE 1
Table of notations.

Symbol	Explanation
$\mathcal{S}$	set of computing server
$\mathcal{R}$	set of computing resources of each server
$\hat{\mathcal{R}}$	augmented set of resources, *i.e.*, $\mathcal{R} \cup \{\text{er}\}$
$\mathcal{J}$	set of users
$c_{s,r}$	capacity of server s for computational resource $r \in \mathcal{R}$
$\mathbf{c}_s$	capacity profile of server s, *i.e.*, $(c_{s,r})_{r\in\mathcal{R}}$
$\mathbf{c}$	computational capacity profile, *i.e.*, $(\mathbf{c}_s)_{s\in\mathcal{S}}$
c^{er}	capacity of the external resource
n_j	number of tasks submitted by user j
$d_{j,r}$	share of resource $r \in \hat{\mathcal{R}}$ required by user j per task execution
$\mathbf{d}_j$	demand profile of user j, *i.e.*, $(d_{j,r})_{r\in\hat{\mathcal{R}}}$
$u_j(.)$	utility function of user j
$\mathbf{u}$	utility profile of the users, *i.e.*, $(u_j)_{j\in\mathcal{J}}$
$E = \left(\mathcal{J}, \mathcal{S}, \hat{\mathcal{R}}, \mathbf{u}, \mathbf{c}, c^{\text{er}}\right)$	environment E with $\mathcal{J}, \mathcal{S}, \hat{\mathcal{R}}, \mathbf{u}, \mathbf{c}$, and c^{er} as its set of users, set of computing server, augmented set of resources, utility profile of the users, computational capacity profile, and capacity of the external resource, respectively
$\mathcal{X}(E)$	set of feasible task allocations for environment E
$\lambda_{j,s}(E)$	number of tasks allocated to user j in server s by applying the non-wasteful allocation rule λ to environment E
$\Lambda_{j,s,r}(E)$	share of computational resource $r \in \mathcal{R}$ that is allocated to user j in server s by applying the non-wasteful allocation rule λ to environment E
$\Lambda^{\text{er}}_j(E)$	share of external resource that is allocated to user j by applying the non-wasteful allocation rule λ to environment E
$\mathbf{\Lambda}_j(E)$	resource allocation profile of user j derived by rule λ for environment E, *i.e.*, $\left(\left((\Lambda_{j,s,r}(E))_{r\in\mathcal{R}}\right)_{s\in\mathcal{S}}, \Lambda^{\text{er}}_j(E)\right)$
η_j	maximum number of tasks that user j can execute by monopolizing the resources if it had submitted unlimited number of tasks, *i.e.*, $\min\left\{\sum_{s\in\mathcal{S}} \underset{r\in\mathcal{R}}{\min}\left\{\frac{c_{s,r}}{d_{j,r}}\right\}, \frac{1}{d_{j,\text{er}}}\right\}$

Note that η_j is not bounded by n_j. For any environment E, the relative utilitarian allocation rule finds the feasible allocation that maximizes the sum of normalized utilities, *i.e.*,

$$\text{arg max}\left\{\sum_{j\in\mathcal{J}} \frac{\sum_{s\in\mathcal{S}} x_{j,s}}{\eta_j} \mid \mathbf{x} \in \mathcal{X}(E)\right\}.$$

Consequently, the relative utilitarian solution is scale-invariant to any affine transformation of the utility function. Moreover, it always remains on the Pareto frontier of the feasibility set. Hence it satisfies PO. However, the relative utilitarian solution violates EF and SI. We consider the Cautious Relative Utilitarian allocation rule (CRU), which maximizes the sum of normalized utilities while maintaining envy-freeness and sharing incentive, *i.e.*, arg max $\left\{ \sum_{j \in \mathcal{J}} \frac{\sum_{s \in \mathcal{S}} x_{j,s}}{\eta_j} \mid \mathbf{x} \in \mathcal{X}(E) \cap \mathcal{X}_{\text{EF}}(E) \cap \mathcal{X}_{\text{SI}}(E) \right\}$, where $\mathcal{X}_{\text{EF}}(E)$ and $\mathcal{X}_{\text{SI}}(E)$ denote the sets of task allocations that satisfy EF and SI for the environment E, respectively. When the number of tasks submitted by the users are unbounded (*i.e.*, $n_j = \infty$ for any user j), $\mathcal{X}(E) \cap \mathcal{X}_{\text{EF}}(E) \cap \mathcal{X}_{\text{SI}}(E)$ can be represented by $|\mathcal{J}|^2 + |\mathcal{S}||\mathcal{R}| + 1$ linear inequalities. Therefore, the CRU solution can be derived efficiently by solving a convex linear programming problem with $|\mathcal{J}||\mathcal{S}|$ decision variables and $|\mathcal{J}|^2 + |\mathcal{S}||\mathcal{R}| + 1$ constraints. Consider the example in Fig. 2. The allocation derived by CRU is illustrated in Fig. 2(a), in which server 1 and server 2 are allocated to user 2 and user 1, respectively.

The CRU allocation always exists (since equal division of resources of each server and the external resource is always a feasible allocation that satisfies EF and SI), and it trivially satisfies EF and SI. Note that the CRU allocation always lies on the Pareto frontier of the set $\mathcal{X}(E) \cap \mathcal{X}_{\text{EF}}(E) \cap \mathcal{X}_{\text{SI}}(E)$, but that does not directly imply that it lies on the Pareto frontier of the set $\mathcal{X}(E)$, which is necessary for satisfying PO. The following theorem states that CRU satisfies PO and fails to satisfy SP. The proof is included in Appendix 9.1.

Theorem 1. *When users submit an unlimited number of tasks (i.e., $n_j = \infty$, $\forall j \in \mathcal{J}$), CRU satisfies EF, PO, and SI, but it fails to satisfy SP.*

3.2 Maximizing Nash Social Welfare

The Nash welfare is the geometric mean of the agents' utilities. Maximizing Nash Welfare (MNW) has been independently discovered in different communities, *e.g.*, Proportional Fairness (PF) in the TCP congestion control literature [44], Nash bargaining solution [26], [27], and Competitive Equilibrium from Equal Incomes (CEEI) in the economic literature [8], [45]. An allocation is proportionally fair if compared to any other feasible allocation of utilities, the aggregate proportional change is less than or equal to 0. The PF allocation can be obtained as the solution to the Nash welfare maximization problem if the feasible utility set is convex [46]. The CEEI would be the competitive equilibrium if all agents were allocated the same budget of some artificial currency. Competitive equilibrium is an allocation vector of resources and a vector of prices, such that i) every agent spends its budget buying an optimal bundle of resources; ii) supply equals demand. Competitive equilibrium is known to guarantee EF and PO [8]. When the resource set is convex and all agents have homogeneous utility functions, *i.e.*, $u_j(\alpha z_j) = \alpha u_j(z_j)$, the CEEI allocation is precisely the same allocation as MNW [47].

In our system model, when the number of tasks submitted by the users is unbounded, the users would have homogeneous utility functions. Thus, the MNW allocation satisfies EF and PO. Moreover, it is easy to show that MNW is scale-invariant to any affine transformation of the utility function. For the example in Fig. 2, MNW gives the same allocation as CRU, which is illustrated in Fig. 2(a). In the following theorem, we observe that MNW satisfies SI and fails to satisfy SP. The proof is included in Appendix 9.2.

Theorem 2. *When users submit an unlimited number of tasks (i.e., $n_j = \infty$, $\forall j \in \mathcal{J}$), MNW satisfies EF, PO, and SI, but it fails to satisfy SP.*

3.3 Egalitarian Approach

Egalitarianism (also known as max-min fairness or Rawlsian rule) maximizes the smallest utility among the agents; subject to that it maximizes the next smallest utility, and so on. Similar to utilitarianism, egalitarianism can be manipulated by applying affine transformations to users' utility functions. The KS allocation rule circumvents this issue by normalizing the utility functions to range from zero to one, then applying the egalitarian allocation rule to the normalized utility functions. While KS trivially satisfies PO and SI for convex feasible utility sets, EF and SP need to be studied carefully for each problem. For instance, while KS satisfies EF and SP for single server cloud computing environments [2], it fails to satisfy them in an MEC environment with multiple access points [48].

Many variants of the egalitarian mechanism for fair resource allocation in a parallel computing environment have been proposed. Contrary to our model, the existing fair allocation mechanisms do not include an external resource in their model. To apply these mechanisms directly, we need to consider the computing servers as meta servers with zero capacity for the external resource and treat the external resource (*e.g.*, the wireless communication link in the MEC example) as a meta server with zero capacity for computing resources. However, existing mechanisms cannot support any task on these servers since the users require both computing resources and link bandwidth, and no meta server contains these resources altogether. Hence, the existing mechanisms are not directly applicable to our problem. Here we study several naive approaches to extend the existing egalitarian mechanisms to our environment, namely dominant resource fairness and task share fairness.

Ghodsi *et al.* proposed DRF to fairly allocate the resources in a cloud computing server [2]. Instead of finding the egalitarian allocation of utilities, DRF applies egalitarianism across the users' share of the dominant resource, which is the resource that the user demands most, *i.e.*, $r_j^* = \arg\max_{r \in \hat{\mathcal{R}}} \{d_{j,r}\}$. For a single server without external resource, DRF satisfies EF, PO, SP, and SI [2]. DRFH generalizes the idea of DRF to environments where resources are pooled by heterogeneous servers [3]. DRFH applies egalitarian fairness across the share of users' dominant resource. It defines the dominant resource as the resource that the user demands most in the artificial server constructed by combining all the heterogeneous servers. Consequently, DRFH's definition of the dominant resource is the same as DRF's. Hence, server heterogeneity does not have any impact on the notion of dominant resource, and this prevents DRFH from satisfying SI [3].

We may naively extend DRFH to our new environment with the external resource by first splitting the external resource across the computing servers and then applying DRFH on this new set of augmented servers with the hope of retaining the properties of DRFH. We denote this naive extension by DRF-$\boldsymbol{\alpha}^{\text{er}}$ where the design parameter $\boldsymbol{\alpha}^{\text{er}} = (\alpha_s^{\text{er}})_{s \in \mathcal{S}}$ specifies the external resource reservation among the servers. Hence, for any environment E, DRF-$\boldsymbol{\alpha}^{\text{er}}$ restricts the feasible task allocation set to

$$\mathcal{X}_{\boldsymbol{\alpha}^{\text{er}}}(E) = \Bigg\{ \mathbf{x} \in \mathbb{R}_+^{|\mathcal{J}| \times |\mathcal{S}|} \mid \sum_{j \in \mathcal{J}} x_{j,s}\, d_{j,\text{er}} \leq \alpha_s^{\text{er}}, \forall s \in \mathcal{S},$$
$$\sum_{j \in \mathcal{J}} x_{j,s}\, d_{j,r} \leq c_{s,r}, \forall r \in \mathcal{R},\, s \in \mathcal{S},$$
$$\sum_{s \in \mathcal{S}} x_{j,s} \leq n_j, \forall j \in \mathcal{J} \Bigg\},$$

and it adopts the same definition for the users' dominant resource as DRF and DRFH. For any environment E, when the users have an unlimited number of submitted tasks (*i.e.*, $n_j = \infty, \forall j \in \mathcal{J}$), DRF-$\boldsymbol{\alpha}^{\text{er}}$ maximizes the equalized dominant share, *i.e.*,

$$\begin{aligned}\max_{g,\, \mathbf{x}} \quad & g \\ \text{s.t.} \quad & \mathbf{x} \in \mathcal{X}_{\boldsymbol{\alpha}^{\text{er}}}(E), \\ & \sum_{s \in \mathcal{S}} x_{j,s} d_{j,r_j^*} = g, \forall j \in \mathcal{J}.\end{aligned}$$

Although DRFH satisfies EF, PO, and SP in the multi-server cloud computing environments, these properties may not be inherited by DRF-$\boldsymbol{\alpha}^{\text{er}}$ in our new environment, which is due to the dissimilarity of the utility functions and the feasibility sets between DRFH and our model. Fixing the external resource reservation among the servers shrinks the task feasibility sets. Consequently, the task allocation derived by DRF-$\boldsymbol{\alpha}^{\text{er}}$, which lies on the Pareto frontier of the shrunken task feasibility set, may not lie on the Pareto frontier of the original task feasibility set. Furthermore, the task allocation that represents the equal division of the resources in the original environment may not be even feasible in the transformed environment.

Consider the example in Fig. 2. The allocation derived by DRF-$\boldsymbol{\alpha}^{\text{er}}$ with $\boldsymbol{\alpha}^{\text{er}} = (1/|\mathcal{S}|)_{s \in \mathcal{S}}$ (*i.e.*, the external resource is equally reserved among the servers) is illustrated in Fig. 2(b). The allocation derived by CRU and MNW, illustrated in Fig. 2(a), shows that both users can execute more tasks when we allocate server 1 and server 2 to user 2 and user 1, respectively. Therefore, PO is violated by this allocation rule. More generally, in the following theorem, we observe that given any heterogeneous server configuration, it is impossible to find an appropriate external resource reservation profile (*i.e.*, $\boldsymbol{\alpha}^{\text{er}}$) for DRF-$\boldsymbol{\alpha}^{\text{er}}$ that guarantees PO or SI. The proof Theorem 3 is included in Appendix 9.3.

Theorem 3. *There exists no feasible external resource reservation profile $\boldsymbol{\alpha}^{er}$, such that DRF-$\boldsymbol{\alpha}^{er}$ satisfies PO or SI.*

TSF was proposed by Wang *et al.* to fairly allocate resources in a heterogeneous multi-resource environment to satisfy all of EF, PO, SP, and SI [33]. TSF admits server heterogeneity and normalizes the number of allocated tasks with respect to the number of tasks that the user can execute when monopolizing these heterogeneous servers. Hence, it does not treat the heterogeneous servers as one aggregate server. TSF then applies egalitarian fairness across the normalized number of allocated tasks.

We may extend TSF in the same way that we extended DRFH. First, we split the communication link across the computing servers, and then we apply TSF on this new set of augmented servers to retain the properties of TSF. We denote this naive extension by TSF-$\boldsymbol{\alpha}^{\text{er}}$. When the users have an unlimited number of submitted tasks, TSF-$\boldsymbol{\alpha}^{\text{er}}$ maximizes the equalized task share, *i.e.*,

$$\max_{g,\, \mathbf{x}} \quad g \tag{7a}$$

$$\text{s.t.} \quad \mathbf{x} \in \mathcal{X}_{\boldsymbol{\alpha}^{\text{er}}}(E), \tag{7b}$$

$$\frac{\sum_{s \in \mathcal{S}} x_{j,s}}{\sum_{s \in \mathcal{S}} \min \left\{ \min_{r \in \mathcal{R}} \left\{ \frac{c_{s,r}}{d_{j,r}} \right\}, \frac{\alpha_s^{\text{er}}}{d_{j,\text{er}}} \right\}} = g, \forall j \in \mathcal{J}. \tag{7c}$$

TABLE 2
Properties of CRU, MNW, DRF-$\boldsymbol{\alpha}^{\text{er}}$, TSF-$\boldsymbol{\alpha}^{\text{er}}$, DRF-ER, and TSF-ER.

	EF	**PO**	**SP**	**SI**
CRU	✓	✓	×	✓
MNW	✓	✓	×	✓
DRF-$\boldsymbol{\alpha}^{\text{er}}$	✓	×	✓	×
TSF-$\boldsymbol{\alpha}^{\text{er}}$	✓	×	✓	×
DRF-ER	✓	✓	✓	×
TSF-ER	✓	✓	✓	✓

For the example in Fig. 2, the allocation derived by TSF-$\boldsymbol{\alpha}^{\text{er}}$ with $\boldsymbol{\alpha}^{\text{er}} = (c_{s,\text{CPU}})_{s \in \mathcal{S}}$ (*i.e.*, the external resource reservation profile follows the same pattern as the servers CPU capacity) is illustrated in Fig. 2(c). The following theorem observes that the properties PO and SI of TSF are not inherited by TSF-$\boldsymbol{\alpha}^{\text{er}}$. For instance, consider the example in Fig. 2. Both users execute a fewer number of tasks with TSF-$\boldsymbol{\alpha}^{\text{er}}$ compared to the allocation they receive from TSF-ER. The proof of Theorem 4 is nearly identical to the proof of Theorem 3 and is omitted.

Theorem 4. *There exists no feasible external resource reservation profile $\boldsymbol{\alpha}^{er}$, such that TSF-$\boldsymbol{\alpha}^{er}$ satisfies PO or SI.*

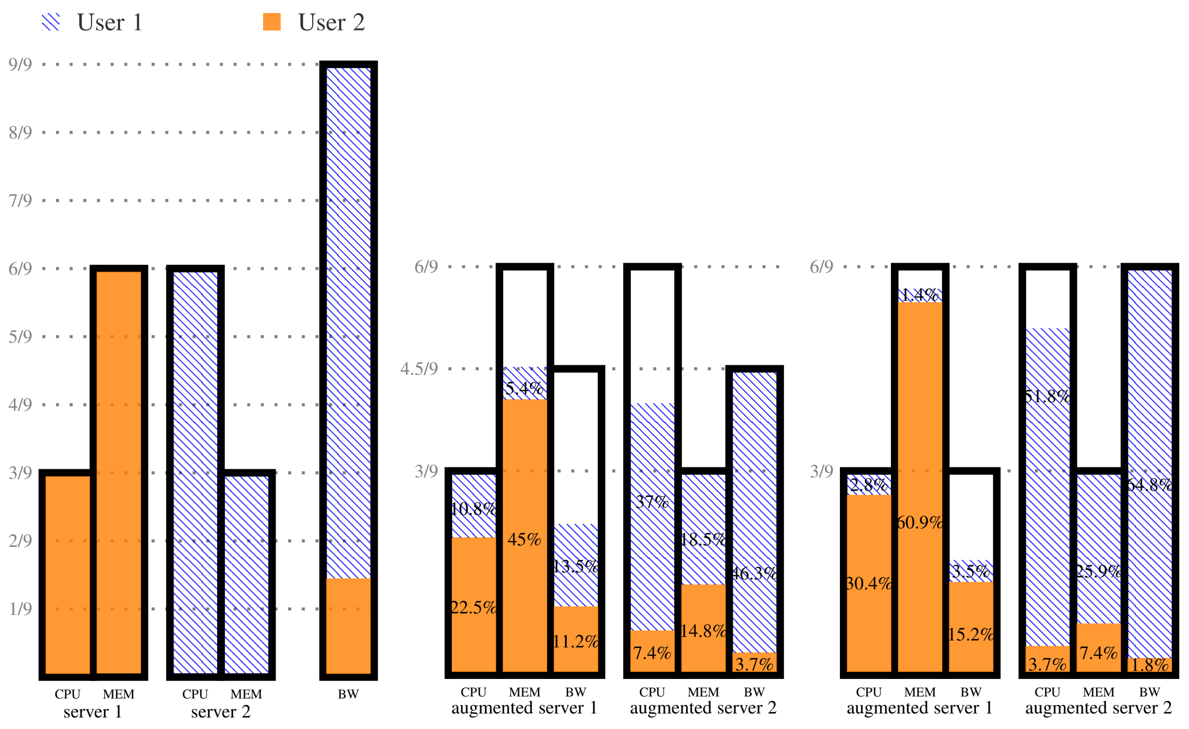
In [37], we proposed DRF-ER, which extends DRF to environments with an external resource. Similar to DRF-$\boldsymbol{\alpha}^{\text{er}}$, it applies egalitarian fairness across the dominant shares with a minor but crucial change. Along with g and $\mathbf{x}$, DRF-ER considers $\boldsymbol{\alpha}^{\text{er}}$ as a decision variable of this problem. Hence, DRF-ER can be interpreted as dynamically selecting the *best* communication link reservation profile that leads to the highest equalized share of dominant resource for the users. In [37], we showed that DRF-ER could satisfy EF, PO, and SP, but fails to satisfy SI. Unfortunately, extending TSF in the same way to dynamically select the best communication link reservation profile leads into a non-convex optimization problem (*i.e.*, problem (7) would be non-convex if we consider $\boldsymbol{\alpha}^{\text{er}}$ to be a decision variable along with g and $\mathbf{x}$). Table 2 summarizes the properties of CRU, MNW, DRF-$\boldsymbol{\alpha}^{\text{er}}$, TSF-$\boldsymbol{\alpha}^{\text{er}}$, and DRF-ER.

4 Task Share Fairness with External Resource

Theorems 3 and 4 indicate that static extension of DRF or TSF does not preserve PO and SI. Furthermore, the dynamic extension of TSF, problem (7), leads to a non-convex optimization problem. In this section, we present TSF-ER, which accommodates an external resource while preserving all of the EF, PO, SP, and SI properties.

Given any environment E, the TSF-ER multi-resource allocation rule applies max-min fairness across the users' normalized number of allocated tasks, *i.e.*, the users' task share. For any user $j \in \mathcal{J}$, the number of allocated tasks is normalized with respect to η_j, which is given in (6) and denotes the maximum number of tasks that user j can execute by monopolizing the servers. Unlike naive extensions of TSF, TSF-ER does not require to split the external resource across the computing servers.

Algorithm 1 gives the pseudo-code of TSF-ER. In the TSF-ER allocation, which is max-min fair across the users' task share, each user $j \in \mathcal{J}$ either belongs to, $\mathcal{J}_{\text{sat}}$, the set of *saturated*



(a) CRU and MNW (b) DRF-$\boldsymbol{\alpha}^{\text{er}}$ with $\boldsymbol{\alpha}^{\text{er}} = (1/|\mathcal{S}|)_{s\in\mathcal{S}}$ (c) TSF-$\boldsymbol{\alpha}^{\text{er}}$ with $\boldsymbol{\alpha}^{\text{er}} = (c_{s,\text{CPU}})_{s\in\mathcal{S}}$

Fig. 2. Illustrative example using MEC with bandwidth (BW) as the external resource. Server 1 contains 5 CPU cores and 10 GB of memory, and server 2 contains 10 CPU cores and 5 GB of memory. To execute a task, user 1 requires 2 CPU cores, 1 GB of memory, and $2.5/15$ link bandwidth, and user 2 requires 1 CPU cores, 2 GB of memory, and $0.5/15$ link bandwidth. Hence, the normalized server capacity profile is $s_1 = (1/3, 2/3), s_2 = (2/3, 1/3)$, and normalized users' demand profiles are $\mathbf{d}_1 = (2/15, 1/15, 2.5/15), \mathbf{d}_2 = (1/15, 2/15, 0.5/15)$. (a) Both CRU and MNW allocate server 1 to user 2 and server 2 to user 1. Each user executes 5 tasks. (b) DRF-eq allocation in which BW is equally reserved for the servers. (c) TSF-cpu in which BW reservation profile follows the same pattern as the servers' CPU capacity.

users, or $\mathcal{J}_{\text{active}}$, the set of *active* users. User j is saturated if it receives n_j tasks. Otherwise, it is an active user and is capable of receiving more tasks. Furthermore, all active users should receive the same normalized number of allocated tasks in the max-min fair allocation across the users' task share.

Given the set of saturated and active users in TSF-ER, the max-min fair allocation across the users' task share can be derived by finding the solution to problem (8). By constraints (8b)-(8d), the users are allocated a non-negative number of tasks on each server, and the resource capacity constraints are not violated. Furthermore, by constraint (8e), the saturated users receive exactly their number of submitted tasks. Finally, by maximizing g in (8a) and constraint (8f), the equalized task share of the active users is maximized. However, the sets of saturated and active users in the TSF-ER allocation are unknown beforehand. Algorithm 1 finds $\mathcal{J}_{\text{sat}}$ and $\mathcal{J}_{\text{active}}$ by updating them in each iteration of the while loop. Initially, all users are considered to be active. In each iteration, given the set of saturated and active users from the previous iteration, Algorithm 1 finds $\mathbf{x}^*$, the solution to problem (8). Next, the for loop of the `TSF-ER` routine checks the allocation to see if any active user has been allocated its number of submitted tasks or more, *i.e.*, the set Q derived by the for loop of the `TSF-ER` routine. The users in Q may have received more than what they can utilize, leading to the violation of Pareto optimality and max-min fairness across the users' task share by the allocation. Thus, these users must be removed from the set of active users and added to the set of saturated users, *i.e.*, lines 12 and 11 of the `TSF-ER` routine. The `TSF-ER` routine repeats this procedure with the updated $\mathcal{J}_{\text{sat}}$ and $\mathcal{J}_{\text{active}}$ until there is no new saturated user. At this point, the `TSF-ER` routine has found the max-min fair allocation across the users' task shares and returns it.

The most computationally intensive step of each iteration of the while loop of Algorithm 1 is finding the solution of the linear program in the LP-TaskShare procedure, which can be done efficiently with the interior point method. We assume that resource capacity and users' demands are rational numbers. Hence, we can represent the constraints of the linear programming problem (8) by the inequality $\mathbf{G}\tilde{\mathbf{x}} \le \mathbf{q}$, where $\tilde{\mathbf{x}}$ is the augmented decision vector with $|\mathcal{J}||\mathcal{S}| + 1$ elements, $\mathbf{G} \in \mathbb{Z}^{m\times(|\mathcal{J}||\mathcal{S}|+1)}$ is an integer-valued matrix where $m = 2|\mathcal{J}| + |\mathcal{J}||\mathcal{S}| + |\mathcal{R}||\mathcal{S}| + 1$, and $\mathbf{q}$ is an integer-valued vector with m elements. Let U denote the maximum absolute value of the elements in $\mathbf{G}$ and $\mathbf{q}$. Under the realistic assumption that the number of users is greater than the number of computing servers and the number of computing resources (*i.e.*, $|\mathcal{S}| < |\mathcal{J}|$ and $|\mathcal{R}| < |\mathcal{J}|$), the computational complexity of the primal-dual path-following interior point method with proper initialization to find an ϵ-optimal solution is $O(|\mathcal{J}|^7 \log \frac{|\mathcal{J}|U}{\epsilon})$ [49]. In each iteration of the while loop, either the procedure terminates at line 10 or at least one new user will be removed from the set of unsaturated users and added to the set of saturated users, *i.e.*, line 11-12. Thus, after at most $|\mathcal{J}|$ iterations, $\mathcal{J}_{\text{active}}$ is empty and the TSF-ER procedure terminates at line 10. Hence, the computational complexity of the TSF-ER procedure is $O(|\mathcal{J}|^8 \log \frac{|\mathcal{J}|U}{\epsilon})$, *i.e.*, polynomial in $|\mathcal{J}|$, $\log U$, and $\log \frac{1}{\epsilon}$.

In the following, we study the properties of TSF-ER. We prove

Algorithm 1: Computing TSF-ER.

```
Input: $E = \left(\mathcal{J}, \mathcal{S}, \hat{\mathcal{R}}, \mathbf{u}, \mathbf{c}, c^{\text{er}}\right)$: The input environment.
Output: $\mathbf{x}^*$: A feasible task allocation for $E$.
Procedure TSF-ER($E$)
1   $\mathcal{J}_{\text{sat}} \leftarrow \emptyset$ ▷The set of saturated users
2   $\mathcal{J}_{\text{active}} \leftarrow \mathcal{J}$ ▷The set of active users
3   while True do
4       $Q \leftarrow \emptyset$
5       $\mathbf{x}^* \leftarrow$ LP-TaskShare($E, \mathcal{J}_{active}, \mathcal{J}_{sat}$)
6       for $j \in \mathcal{J}_{active}$ do
7           if $\sum_{s\in\mathcal{S}} x^*_{j,s} \geq n_j$ then
8               $Q \leftarrow Q \cup \{j\}$ ▷$j$ is saturated
9       if $Q = \emptyset$ then
10          return $\mathbf{x}^*$
        else
11          $\mathcal{J}_{\text{active}} \leftarrow \mathcal{J}_{\text{active}}/Q$
12          $\mathcal{J}_{\text{sat}} \leftarrow \mathcal{J}_{\text{sat}} \cup Q$

Procedure LP-TaskShare($E, \mathcal{J}_{active}, \mathcal{J}_{sat}$)
1   Solve
```

$$\max_{g,\mathbf{x}} \quad g \tag{8a}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{J}}\sum_{s\in\mathcal{S}} x_{j,s}\, d_{j,\text{er}} \leq 1, \tag{8b}$$

$$\sum_{j\in\mathcal{J}} x_{j,s}\, d_{j,r} \leq c_{s,r}, \forall s \in \mathcal{S},\ r \in \mathcal{R}, \tag{8c}$$

$$\mathbf{x} \geq 0, \tag{8d}$$

$$\sum_{s\in\mathcal{S}} x_{j,s} = n_j, \forall j \in \mathcal{J}_{\text{sat}}, \tag{8e}$$

$$\frac{\sum_{s\in\mathcal{S}} x_{j,s}}{\eta_j} = g, \forall j \in \mathcal{J}_{\text{active}}. \tag{8f}$$

```
2   return $\mathbf{x}$
```

that it satisfies EF, PO, SP, and SI. We first propose the following lemma, which will be used to prove Theorems 5 and 6. This lemma indicates that the users that are not saturated by TSF-ER have the maximum normalized number of allocated tasks among all users. We use $\mathcal{J}_{\text{active}}(E;\lambda)$ to denote the set of users who do not meet their task limit when allocation rule λ is applied. The proof of this lemma is included in Appendix 9.4.

Lemma 1. *Let* $E = \left(\mathcal{J}, \mathcal{S}, \hat{\mathcal{R}}, \mathbf{u}, \mathbf{c}, c^{er}\right)$ *be an arbitrary environment and* λ *denote the TSF-ER rule. The users that remain active by the end of Algorithm 1 must have the largest task share,* i.e.*, for any* $j \in \mathcal{J}_{active}(E;\lambda)$ *and* $i \in \mathcal{J}$*,*

$$\frac{\sum_{s\in\mathcal{S}} \lambda_{j,s}(E)}{\eta_j} \geq \frac{\sum_{s\in\mathcal{S}} \lambda_{i,s}(E)}{\eta_i}.$$

Theorem 5. *The TSF-ER rule is max-min fair across the users' task shares.*

Proof. Let λ be the TSF-ER resource allocation rule. Suppose user $j \notin \mathcal{J}_{\text{active}}(E;\lambda)$. The dominant share of this user cannot be increased since it is saturated. Thus, it is only possible to increase the dominant share of the users that are in $\mathcal{J}_{\text{active}}(E;\lambda)$. By Lemma 1, the users in $\mathcal{J}_{\text{active}}(E;\lambda)$ have the largest dominant share. Moreover, the task shares of members of $\mathcal{J}_{\text{active}}(E;\lambda)$ are maximized while being equalized by procedure LP-TaskShare. Hence, it is impossible to increase the dominant share of a user $j \in \mathcal{J}_{\text{active}}(E;\lambda)$ without decreasing the dominant share of other users. Consequently, the TSF-ER allocation rule is max-min fair across the users' task shares. □

Next, we present the following lemma, which finds the number of tasks that user j of environment E can execute if it gets the allocation of user k from environment E'. We use this lemma to prove the envy-freeness of TSF-ER.

Lemma 2. *Let* λ *be an arbitrary non-wasteful allocation rule and* E *and* E' *be two arbitrary environments with the same set of augmented resource* $\hat{\mathcal{R}}$*,* i.e.*,* $E = \left(\mathcal{J}, \mathcal{S}, \hat{\mathcal{R}}, \mathbf{u}, \mathbf{c}, c_{er}\right)$ *and* $E' = \left(\mathcal{J}', \mathcal{S}', \hat{\mathcal{R}}, \mathbf{u}', \mathbf{c}', c'_{er}\right)$*. The number of tasks that user* $j \in \mathcal{J}$ *in environment* E *can execute if it gets the allocation of user* $k \in \mathcal{J}'$ *in environment* E' *is*

$$u_j\left(\mathbf{\Lambda}_k(E')\right) = min\left\{\sum_{s\in\mathcal{S}'} \lambda_{k,s}(E')\ \underset{r\in\hat{\mathcal{R}}}{min}\left\{\frac{d'_{k,r}}{d_{j,r}}\right\}, n_j\right\}. \tag{9}$$

Proof. The combination of (1), (3), and (4) yields Lemma 2. □

In the following theorem we observe the envy-freeness of TSF-ER.

Theorem 6. *The TSF-ER rule satisfies EF.*

Proof. We start by proving the envy-freeness for the saturated users. Let λ be the TSF-ER resource allocation rule and E be an arbitrary environment. Any user $j \notin \mathcal{J}_{\text{active}}(E;\lambda)$ must have been saturated by Algorithm 1 and received the maximum number of tasks that it can execute. Hence, it cannot envy any other user's allocation.

Next, we prove the envy-freeness for the unsaturated users. Let $j \in \mathcal{J}_{\text{active}}(E;\lambda)$ be any arbitrary unsaturated user and $i \in \mathcal{J}$ be any arbitrary user. By Lemma 2, the number of tasks that user j can execute with user i's allocated resource, *i.e.*, $u_j(\mathbf{\Lambda}_i(E))$, is no more than $\sum_{s\in\mathcal{S}} \lambda_{i,s}(E)\ \min_{r\in\hat{\mathcal{R}}}\left\{\frac{d_{i,r}}{d_{j,r}}\right\}$. Furthermore, by Lemma 1, $\sum_{s\in\mathcal{S}} \lambda_{i,s}(E) \leq \frac{\eta_i \sum_{s\in\mathcal{S}} \lambda_{j,s}(E)}{\eta_j}$. Hence, the number of tasks that user j can execute with user i's allocation is no more than $\sum_{s\in\mathcal{S}} \lambda_{j,s}(E)\ \min_{r\in\hat{\mathcal{R}}}\left\{\frac{d_{i,r}\eta_i}{d_{j,r}\eta_j}\right\}$, *i.e.*,

$$u_j\left(\mathbf{\Lambda}_i(E)\right) \leq \sum_{s\in\mathcal{S}} \lambda_{j,s}(E)\ \min_{r\in\hat{\mathcal{R}}}\left\{\frac{d_{i,r}\eta_i}{d_{j,r}\eta_j}\right\}.$$

It remains to show that $\min_{r\in\hat{\mathcal{R}}}\left\{\frac{d_{i,r}\eta_i}{d_{j,r}\eta_j}\right\} \leq 1$.

We consider the following two cases and show that the inequality holds in both cases.

Case 1. $\eta_j = \frac{1}{d_{j,\text{er}}}$:

$$\min_{r\in\hat{\mathcal{R}}}\left\{\frac{d_{i,r}\eta_i}{d_{j,r}\eta_j}\right\} = \min_{r\in\hat{\mathcal{R}}}\left\{\frac{d_{i,r}\eta_i}{d_{j,r}/d_{j,\text{er}}}\right\} \leq \frac{d_{i,\text{er}}\eta_i}{d_{j,\text{er}}/d_{j,\text{er}}} \leq \frac{d_{i,\text{er}}}{d_{i,\text{er}}} = 1.$$

Case 2. $\eta_j = \sum\limits_{s\in\mathcal{S}} \min\limits_{r\in\mathcal{R}} \left\{\frac{c_{s,r}}{d_{j,r}}\right\}$:

$$\begin{aligned}\min_{r\in\hat{\mathcal{R}}}\left\{\frac{d_{i,r}\eta_i}{d_{j,r}\eta_j}\right\} &= \frac{\min\limits_{r\in\hat{\mathcal{R}}}\left\{\frac{d_{i,r}}{d_{j,r}}\right\}\eta_i}{\sum\limits_{s\in\mathcal{S}}\min\limits_{r\in\mathcal{R}}\left\{\frac{c_{s,r}}{d_{j,r}}\right\}} \\ &\le \frac{\min\limits_{r\in\hat{\mathcal{R}}}\left\{\frac{d_{i,r}}{d_{j,r}}\right\}\sum\limits_{s\in\mathcal{S}}\min\limits_{r\in\mathcal{R}}\left\{\frac{c_{s,r}}{d_{i,r}}\right\}}{\sum\limits_{s\in\mathcal{S}}\min\limits_{r\in\mathcal{R}}\left\{\frac{c_{s,r}}{d_{j,r}}\right\}} \\ &\le \frac{\min\limits_{r\in\mathcal{R}}\left\{\frac{d_{i,r}}{d_{j,r}}\right\}\sum\limits_{s\in\mathcal{S}}\min\limits_{r\in\mathcal{R}}\left\{\frac{c_{s,r}}{d_{i,r}}\right\}}{\sum\limits_{s\in\mathcal{S}}\min\limits_{r\in\mathcal{R}}\left\{\frac{c_{s,r}}{d_{j,r}}\right\}} \\ &\le \frac{\sum\limits_{s\in\mathcal{S}}\min\limits_{r\in\mathcal{R}}\left\{\frac{d_{i,r}}{d_{j,r}}\right\}\min\limits_{r\in\mathcal{R}}\left\{\frac{c_{s,r}}{d_{i,r}}\right\}}{\sum\limits_{s\in\mathcal{S}}\min\limits_{r\in\mathcal{R}}\left\{\frac{c_{s,r}}{d_{j,r}}\right\}} \\ &\le \frac{\sum\limits_{s\in\mathcal{S}}\min\limits_{r\in\mathcal{R}}\left\{\frac{d_{i,r}}{d_{j,r}}\times\frac{c_{s,r}}{d_{i,r}}\right\}}{\sum\limits_{s\in\mathcal{S}}\min\limits_{r\in\mathcal{R}}\left\{\frac{c_{s,r}}{d_{j,r}}\right\}} \\ &= \frac{\sum\limits_{s\in\mathcal{S}}\min\limits_{r\in\mathcal{R}}\left\{\frac{c_{s,r}}{d_{j,r}}\right\}}{\sum\limits_{s\in\mathcal{S}}\min\limits_{r\in\mathcal{R}}\left\{\frac{c_{s,r}}{d_{j,r}}\right\}} = 1.\end{aligned}$$

This concludes the proof of Theorem 6. □

In the following theorem we observe the Pareto-optimality of the TSF-ER allocation.

Theorem 7. *The TSF-ER rule satisfies PO.*

Proof. Let λ be the TSF-ER resource allocation rule. Assume, by way of contradiction, that allocation obtained by λ does not satisfy PO. Thus, there exists an environment $E = \left(\mathcal{J}, \mathcal{S}, \hat{\mathcal{R}}, \mathbf{u}, \mathbf{c}, c^{\text{er}}\right)$ and a feasible task allocation $\mathbf{x} \in \mathcal{X}(E)$ for this environment such that $\sum_{s\in\mathcal{S}} x_{i,s} = \sum_{s\in\mathcal{S}} \lambda_{i,s}(E)$ for any $i \notin \mathcal{J}_{\text{active}}(E;\lambda)$, $\sum_{s\in\mathcal{S}} x_{i,s} \ge \sum_{s\in\mathcal{S}} \lambda_{i,s}(E)$ for any $i \in \mathcal{J}_{\text{active}}(E;\lambda)$, and there exists some user $j \in \mathcal{J}_{\text{active}}(E;\lambda)$ with strict inequality. We use $\mathbf{x}$ to construct $\mathbf{x}' \in \mathcal{X}(E)$ such that

$$\sum_{s\in\mathcal{S}} x'_{i,s}/\eta_i = \sum_{s\in\mathcal{S}} \lambda_{i,s}(E)/\eta_i,\ \forall i \notin \mathcal{J}_{\text{active}}(E;\lambda), \quad (10)$$

$$\sum_{s\in\mathcal{S}} x'_{i,s}/\eta_i > \sum_{s\in\mathcal{S}} \lambda_{i,s}(E)/\eta_i,\ \forall i \in \mathcal{J}_{\text{active}}(E;\lambda). \quad (11)$$

This contradicts the premise of max-min fairness of TSF-ER allocation across the users' normalized number of allocated tasks. In what follows, we show how to construct $\mathbf{x}'$.

Consider the feasible allocation $\mathbf{x} \in \mathcal{X}(E)$ and the user $j \in \mathcal{J}_{\text{active}}(E;\lambda)$ with $\sum_{s\in\mathcal{S}} x_{i,s} > \sum_{s\in\mathcal{S}} \lambda_{i,s}(E)$. There exists some $\delta > 0$ such that $\sum_{s\in\mathcal{S}} x_{j,s} \ge \sum_{s\in\mathcal{S}} \lambda_{j,s}(E) + |\mathcal{S}|\delta$. Hence, there exists some server s_0 such that $x_{j,s_0} \ge \lambda_{j,s_0}(E) + \delta$. We construct $\mathbf{x}'$ by reducing δ tasks from user j in server s_0 and adding $\min\left\{\min\limits_{r\in\hat{\mathcal{R}}}\left\{\frac{\delta d_{j,r}}{\sum\limits_{k\in\mathcal{J}_{\text{active}}(E;\lambda)} d_{k,r}}\right\}, n_k - \sum\limits_{s\in\mathcal{S}} x_{k,s}\right\}$ tasks to each user k in $\mathcal{J}_{\text{active}}(E;\lambda)$, including user j, in server s_0. It is easy to check that (10) and (11) hold and $\mathbf{x}' \in \mathcal{X}(E)$. □

Now, we propose the following two lemmas, which will be used to prove that TSF-ER satisfies SP. In the following two lemmas and Theorem 8, we let $E = \left(\mathcal{J}, \mathcal{S}, \hat{\mathcal{R}}, \mathbf{u}, \mathbf{c}, c^{\text{er}}\right)$ be an arbitrary environment in which user j reports its true demand, *i.e.*, $\mathbf{d}$, and $E' = \left(\mathcal{J}, \mathcal{S}, \hat{\mathcal{R}}, \mathbf{u}', \mathbf{c}, c^{\text{er}}\right)$ be the environment in which user j reports some fake demand, *i.e.*, $\mathbf{d}' \ne \mathbf{d}$. Furthermore, we denote the true maximum number of tasks that any arbitrary user $i \in \mathcal{J}$ can execute when it monopolizes the servers in environments E by η_i (*i.e.*, (6)), and denote the fake maximum number of tasks that any arbitrary user $i \in \mathcal{J}$ can execute when it monopolizes the servers in environments E' by η'_i, *i.e.*, $\eta'_i = \min\left\{\sum\limits_{s\in\mathcal{S}}\min\limits_{r\in\mathcal{R}}\left\{\frac{c_{s,r}}{d'_{i,r}}\right\}, \frac{1}{d'_{i,\text{er}}}\right\}$. Note that $\eta'_i = \eta_i$ for all $i \ne j$.

By Lemma 3, a user can benefit from reporting a fake demand only if its resulting normalized number of allocated tasks is larger than its original normalized number of allocated tasks when the user reports its true demand. The proof of Lemma 3 is included in Appendix 9.5.

Lemma 3. *Let E, j, E', η_i, and η'_i be as defined above. If the normalized number of tasks allocated to user j is not increased in environment E', i.e., $\sum_{s\in\mathcal{S}} \lambda_{j,s}(E')/\eta'_j \le \sum_{s\in\mathcal{S}} \lambda_{j,s}(E)/\eta_j$, it does not benefit from the reported fake demand, i.e., $u_j(\Lambda_j(E')) \le u_j(\Lambda_j(E))$.*

In Lemma 4 we study how the users' normalized number of allocated tasks change if a user could benefit from reporting a fake demand. Consider two users j and k in the environment E such that user k's normalized number of allocated tasks is not greater than that of user j. Let us assume that user j reports some fake demand and increased its normalized number of allocated tasks in this new environment E'. We show that user k's normalized number of allocated tasks in environment E' cannot be smaller than its normalized number of allocated tasks in environment E. The proof of Lemma 4 is included in Appendix 9.6.

Lemma 4. *Let E, j, E', η_i, and η'_i be as defined above. Let $k \ne j$ be a user such that $\sum_{s\in\mathcal{S}} \lambda_{j,s}(E)/\eta_j \ge \sum_{s\in\mathcal{S}} \lambda_{k,s}(E)/\eta_k$. If the normalized number of allocated tasks to user j is increased after reporting a fake demand, i.e., $\sum_{s\in\mathcal{S}} \lambda_{j,s}(E')/\eta'_j > \sum_{s\in\mathcal{S}} \lambda_{j,s}(E)/\eta_j$, user k's number of allocated tasks cannot decrease, i.e., $\sum_{s\in\mathcal{S}} \lambda_{k,s}(E') \ge \sum_{s\in\mathcal{S}} \lambda_{k,s}(E)$.*

Theorem 8. *The TSF-ER rule satisfies SP.*

Proof. Note that a user's number of submitted tasks does not have any impact on η_j and can only bound the number of tasks it may receive. Hence, the users cannot benefit from reporting a fake number of tasks. It remains to show that the users cannot benefit from reporting a fake demand. Let E, j, E', η_i, and η'_i be as defined above. If $j \notin \mathcal{J}_{\text{active}}(E;\lambda)$, user j cannot benefit from reporting fake demand, since it already receives the maximum number of tasks that it can execute, *i.e.*, n_j. It remains to show that user j cannot benefit from reporting fake demand even if $j \in \mathcal{J}_{\text{active}}(E;\lambda)$. We prove it separately for the cases of $\sum_{s\in\mathcal{S}} \lambda_{j,s}(E')/\eta'_j \le \sum_{s\in\mathcal{S}} \lambda_{j,s}(E)/\eta_j$ and $\sum_{s\in\mathcal{S}} \lambda_{j,s}(E')/\eta'_j > \sum_{s\in\mathcal{S}} \lambda_{j,s}(E)/\eta_j$. In the first case, Lemma 3 implies that user j cannot benefit from the reported fake demand. In the latter case, since $j \in \mathcal{J}_{\text{active}}(E;\lambda)$, we have $\sum_{s\in\mathcal{S}} \lambda_{j,s}(E)/\eta_j \ge \sum_{s\in\mathcal{S}} \lambda_{k,s}(E)/\eta_k$ for any $k \ne j$. Hence, by Lemma 4, $\sum_{s\in\mathcal{S}} \lambda_{k,s}(E') \ge \sum_{s\in\mathcal{S}} \lambda_{k,s}(E)$, *i.e.*, the number of tasks allocated to any user $k \ne j$ does not decrease in the new environment. If j executes more tasks with the resources it receives in the new environment E', we have found a feasible allocation in which j is strictly better off and no user is worsen off. This contradicts Pareto optimality of the TSF-ER rule. Hence, j cannot execute more tasks with the resources it receives after reporting the fake demand, and TSF-ER satisfies SP. □

Finally, we show that TSF-ER satisfies SI.

Theorem 9. *The TSF-ER rule satisfies SI.*

Proof. It suffices to show that TSF-ER does not allocate fewer than $1/|\mathcal{J}|$ tasks to each user in the worst case scenario where all users submitted an unlimited number of tasks. Consider the feasible allocation of equally sharing each server and the external resource among all users. Then, the normalized number of allocated tasks for any user j is $1/|\mathcal{J}|$. Problem (8) maximizes the equalized normalized number of allocated tasks. Hence, its solution cannot be worse than $1/|\mathcal{J}|$. Therefore, TSF-ER satisfies SI. □

5 Online Implementation of TSF-ER with Indivisible Tasks

In Sec. 4, we have studied the properties of TSF-ER. These properties are defined for settings with divisible tasks, where the allocation is recomputed whenever a user arrives or departs. Under task indivisibility, TSF-ER can be implemented using a simple online algorithm. We give the pseudo-code of this online implementation of TSF-ER in Algorithm 2.

At the beginning of each reallocation time slot, TSF-ER runs the `Reallocation` routine, which suspends all the running tasks, adds them back to the waiting queue, resets the users' fair share, and releases all of the allocated resources. Next, it calls the `AllocateFromQueues` routine. In each iteration, it allocates one task of the user that is the furthest from its fair share (*i.e.*, the most disadvantaged user) and assigns it to the server that fits that user's demand best (line 6 of the `AllocateFromQueues` routine). In particular, this implementation of TSF-ER for indivisible tasks serves users in ascending order of task shares. If multiple users have the minimum task share, it chooses the user with the minimum task share that produces the minimum increase in its task share after allocating one task, *i.e.*, the user with minimum $1/\eta_j$. The algorithm keeps allocating the user with the least task share until no more tasks can be scheduled, *i.e.*, $\mathcal{J}^{\text{feas}} = \emptyset$. At this point, the scheduler stops and waits for the next reallocation time slot.

Upon task arrival in each reallocation time slot, TSF-ER runs the `TaskArrival` routine. The arrived tasks are queued up to be scheduled by the `AllocateFromQueues` routine in line 5 of the `TaskArrival` routine. Note that the `TaskArrival` routine does not revoke any of the tasks that are already running on the machines, and it allocates tasks if there exist enough available resources for them.

Upon task completion in each reallocation time slot, TSF-ER runs the `TaskDeparture` routine. It removes the completed tasks from their allocated machines and releases the allocated resources accordingly. Finally, it calls the `AllocateFromQueues` routine to see if the released resources are enough to allocate any tasks in the waiting queue.

If no task arrival or departure has occurred during a reallocation time slot, there is no need to reallocate the tasks at the beginning of the next time slot and the `Reallocation` call will be ignored. In designing the reallocation time slot length, the overhead caused by the suspend/resume procedure must be considered. In the next section, we will study the trade-off between the reallocation time slot length and job completion time.

Algorithm 2: Computing TSF-ER allocation under task indivisibility.

```
Input: $\mathcal{J}^{\text{arr}}$: Set of arrived users.
       $\mathcal{I}_j$: Set of arrived tasks of user $j$.
Procedure TaskArrival($\mathcal{J}^{arr}$, $(\mathcal{I}_j)_{j\in\mathcal{J}^{arr}}$)
1   for $j \in \mathcal{J}^{arr}$ do
2       Create a waiting task queue for the arrived user $j$
3       for $i \in \mathcal{I}_j$ do
4           Append the arrived task $i$ to the waiting task queue of user $j$
5   AllocateFromQueues()

Input: $\mathcal{J}^{\text{dep}}$: Set of departing users.
       $\mathcal{I}_j$: Set of departing tasks of user $j$.
Procedure TaskDeparture($\mathcal{J}^{dep}$, $(\mathcal{I}_j)_{j\in\mathcal{J}^{dep}}$)
1   for $j \in \mathcal{J}^{dep}$ do
2       for $i \in \mathcal{I}_j$ do
3           Remove task $i$ from its allocated machine and release its allocated resources
4           $s_j \leftarrow s_j - \frac{1}{\eta_j}$ ▷ decrease user $j$'s active task share by one task share
5   AllocateFromQueues()

Procedure Reallocation()
1   Suspend all running tasks and add them back to their corresponding waiting task queues
2   Set the active task share of all users to zero
3   Release the allocated resources
4   AllocateFromQueues()

Procedure AllocateFromQueues()
1   $\mathcal{J}^{\text{feas}} \leftarrow$ the set of users with feasible tasks in their waiting task queue with respect to the available resources
2   while $\mathcal{J}^{feas} \neq \emptyset$ do
3       $\mathcal{J}^{\text{min}} \leftarrow \arg\min_{j\in\mathcal{J}^{\text{feas}}} \{s_j\}$ ▷ $s_j$ is the active task share of user $j$
4       $j^* \leftarrow \arg\min_{j\in\mathcal{J}^{\text{min}}} \left\{\frac{1}{\eta_j}\right\}$ ▷ select the user with the minimum one task share
5       $\mathcal{S}_{j^*} \leftarrow$ the set of feasible machines for the selected task
6       $m \leftarrow \arg\min_{s\in\mathcal{S}_{j^*}} \{\sum_{r\in\mathcal{R}}(a_{s,r} - d_{j^*,r})^2\}$ ▷ find the machine that fits user $j^*$'s demand
7       Select one task from user $j^*$'s waiting task queue and assign it to machine $m$
8       Update the available resources
9       $s_{j^*} \leftarrow s_{j^*} + \frac{1}{\eta_{j^*}}$ ▷ increase user $j^*$'s active task share by one task share
10      $\mathcal{J}^{\text{feas}} \leftarrow$ the set of users with feasible tasks in their waiting task queue with respect to the available resources
```

The `AllocateFromQueues` call is the most computationally intensive part of the `TaskArrival`, `TaskDeparture`, and `Reallocation` routines. Due to line 10 of the

`AllocateFromQueues` routine, each iteration of the while loop of this procedure has the computational complexity of $O(|\mathcal{J}||\mathcal{S}||\mathcal{R}|)$. Let $n = \sum_{j\in\mathcal{J}} n_j$ denote the total number of submitted tasks. The number of iterations of the while loop is bounded by n. Hence, the computational complexity of the `AllocateFromQueues` routine is $O(n^2|\mathcal{S}||\mathcal{R}|)$.

6 Experimental Results

As summarized in Table 2, unlike the existing allocation rules, TSF-ER algorithmically guarantees all of the fairness and efficiency properties. This section further evaluates its performance in job completion time, resource utilization, and fairness. First, we evaluate the performance of TSF-ER, MNW, and CRU under the fluid task model via Google cluster trace [50] with three-server clusters. Next, we study their performance under task indivisibility via large-scale simulation using Alibaba trace [51]. We emphasize that TSF-ER is not explicitly designed to optimize the users' job completion time or resource utilization. Instead, the main benefit of TSF-ER is that it simultaneously satisfies all four fairness and efficiency properties EF, PO, SP, and SI.

6.1 Simulation Process

This section explains our simulation process under two different models for users' tasks, namely the fluid and indivisible task models. In both models, the users submit some number of tasks for execution and wait for their tasks to be executed by the servers and leave the system afterward. A user's job completion time (JCT) is the time between its arrival and finishing all of its tasks.

Fluid task model: We assume that the tasks are infinitesimally divisible and can be stopped, reallocated, and resumed anytime without any overhead. Hence, upon arrival or departure of any task, the scheduler stops all the running tasks, reallocates them as per the allocation rule, and resumes their execution. Recall that $x_{j,s}$ is the number of tasks that is allocated to user j on server s. We interpret the total number of tasks allocated to user j in any interval, *i.e.*, $\sum_{s\in\mathcal{S}} x_{j,s}$, as the processing rate of user j. User j would leave the system when all its tasks are executed. The user's JCT is defined as the time between its arrival and departure. Note that in the fluid model, we do not bound the processing rate and allow the scheduler to allocate even more than the number of submitted tasks.

Indivisible task model: We assume that the scheduler is not allowed to allocate tasks fractionally, and it cannot allocate more than the number of submitted tasks. Similar to [52] and [53], we consider an overhead of 250 ms for suspend-resume caused by the reallocation procedure. This overhead is due to suspending a task and storing the task state for future allocation. The scheduling procedure of TSF-ER is described in Algorithm 2. However, extending MNW and CRU to the indivisible task model is not straightforward and is beyond the scope of this paper. Here we implement them by suspending all running tasks at each reallocation time slot and finding the MNW and CRU fractional allocation for the fluid task model. Next, the fractional allocation is rounded down to suit the indivisible task model. Upon arrival or departure of some users during each reallocation time slot, the scheduler keeps allocating tasks to the server randomly until no more tasks can be allocated.

6.2 Small-scale Experiments with Google Cluster Trace

In this section, we evaluate the performance of the allocation rules under fluid and indivisible task models with a small number of users dynamically arriving in the system with three machines and an external resource. This small-scale simulation setting allows us to run the experiment for over 10K resource allocation problem instances and study the effect of various parameters, namely the number of arriving users, server heterogeneity, external resource capacity, and reallocation slot length, on the users' JCT. We are not aware of any publicly available MEC datasets in which users require multiple resources and a resource external to them, which also specifies the users' demand for each resource. Therefore, we use Google cluster trace datasets to simulate an MEC environment and synthesize users' demand for bandwidth (*i.e.*, the external resource) from their CPU demands. In what follows, we explain our simulation settings and introduce the notion of JCT factor.

User configuration: The Google cluster trace describes every job submission and its resource usage in the Google Borg compute cluster for a month. Each job is divided into tasks with the same resource demands, and the jobs' arrival time, the number of submitted tasks, per task execution time, and per task CPU and memory demands are included in the trace. For simplicity, we consider each job in the trace as a distinct user in our simulation setting. Thus, in what follows, we use job and user interchangeably. Similar to [54] - [57], we synthesize the users' required bit rate by assuming that $d_{j,\text{CPU}} f_{\text{CPU}} = X d_{j,\text{bps}}$, where $d_{j,\text{CPU}}$ is the CPU demand of user j, f_{CPU} is the CPU frequency of the servers, $d_{j,\text{bps}}$ is the required bit rate of user j, and X is a random variable with Gamma distribution. We set the CPU frequency to be 2.6 GHz and follow [54] to set the shape and rate parameter of X to 4 and 200, respectively. We consider a general MEC system with frequency division multiple access with spectral efficiency of 3.5 bit/s/Hz and find user j's demand for the external resource, *i.e.*, bandwidth, as $d_{j,\text{er}} = d_{j,\text{bps}}/3.5$. In each resource allocation problem instance, we set the number of users that dynamically arrive to the MEC environment to 10, 15, or 20. In each resource allocation problem instance, the users are uniformly sampled from the first 1000 job submissions in Google cluster trace, which span over the first 48 hours of the Google cluster trace.

Server and external resource configuration: In the Google cluster trace, the tasks' demand and the servers' capacity for each resource are scaled to make the maximum capacity of each resource in the servers of the Google Borg compute cluster equal to 1. We pick the server with the capacity of $(1, 1)$ from the Google cluster trace, where the first and the second entry denote the scaled capacity for CPU cores and memory, and add two synthesized servers with the capacity of $(1 + \rho, 1 - \rho)$, and $(1-\rho, 1+\rho)$, where ρ represents the level of heterogeneity among the servers, while keeping the aggregate capacity of each resource equal to 3. In particular, in each resource allocation problem instance, we set ρ to 0.5, 0.7, or 0.9 to study three levels of server heterogeneity. For the external resource, we set the BW of the wireless communication channel to 1, 2, and 3 MHz to study the sensitivity of the four allocation rules to the external resource capacity. Our simulation results show that having a BW of larger than 3 MHz makes BW irrelevant due to the absence of competition over this resource.

JCT factor: The first 1000 job submissions in the Google cluster trace contain jobs with per task execution time from 15

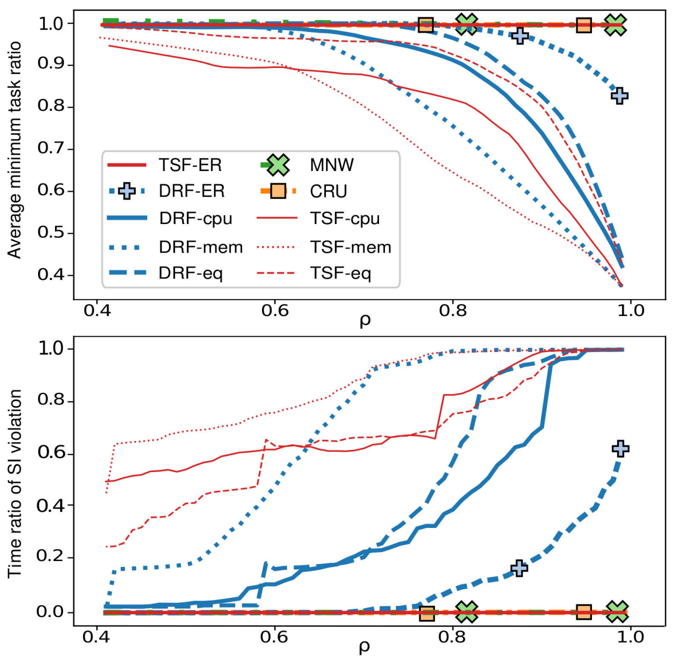


Fig. 3. SI violation vs. server heterogeneity. DRF-$\boldsymbol{\alpha}^{\text{er}}$, TSF-$\boldsymbol{\alpha}^{\text{er}}$, and DRF-ER are further away from satisfying SI and violate SI more often when the server heterogeneity increases.

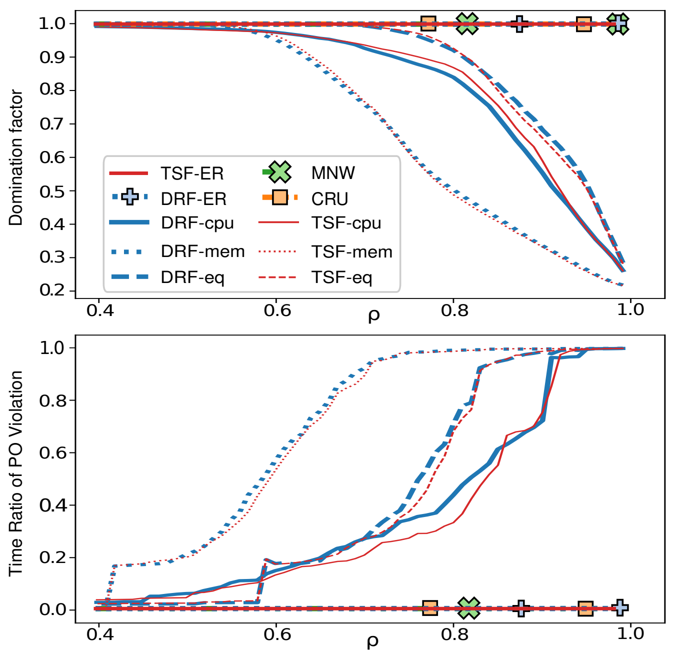


Fig. 4. PO violation vs. server heterogeneity. DRF-$\boldsymbol{\alpha}^{\text{er}}$ and TSF-$\boldsymbol{\alpha}^{\text{er}}$ are less Pareto efficient and more often violate PO when the server heterogeneity increases.

seconds to 28 days. Moreover, the number of tasks that jobs submit varies from 1 to 2854, and their requested resources could be drastically different. Hence, instead of a user's JCT under an allocation rule, we study the *JCT factor*, that is the ratio of the user's standalone JCT to the user's JCT. A user's standalone JCT is defined as its JCT if it had received all resources of all servers. In Sec. 6.2.1 and 6.2.2, the users' standalone JCT is calculated based on the fluid task model and the indivisible task model, respectively. Note that the users' JCT factor is bounded by 1, and the larger it is, the happier the user is with its resource allocation experience.

Before presenting our main simulation results, we study the performance of naive extension of DRFH and TSF, namely DRF-$\boldsymbol{\alpha}^{\text{er}}$ and TSF-$\boldsymbol{\alpha}^{\text{er}}$, and DRF-ER, and argue that TSF-ER dominates them. Theorems 3 and 4 show that there always exists some environment in which DRF-$\boldsymbol{\alpha}^{\text{er}}$ and TSF-$\boldsymbol{\alpha}^{\text{er}}$ fail to find an allocation that provides PO or SI. To see how often these allocation rules violate these properties in realistic environments, we evaluate their performance using the Google cluster trace. We use DRF-cpu (resp. TSF-cpu), DRF-mem (resp. TSF-mem), and DRF-eq (resp. TSF-eq) to denote the three specific design options of DRF-$\boldsymbol{\alpha}^{\text{er}}$ (resp. TSF-$\boldsymbol{\alpha}^{\text{er}}$) when $\boldsymbol{\alpha}^{\text{er}}$ is equal to $(c_{s,\text{CPU}})_{s\in\mathcal{S}}$, $(c_{s,\text{mem}})_{s\in\mathcal{S}}$, and $(1/|\mathcal{S}|)_{s\in\mathcal{S}}$, respectively. Thus, the external resource reservation profile of DRF-cpu (resp. DRF-mem) follows the same pattern as the servers CPU (resp. memory) capacity, and DRF-eq corresponds to the case where the external resource is equally reserved among the servers.

Given $x^*_{j,s}$, an allocation rule's number of allocated tasks to user j on server s, and the total number of tasks that user j can execute when the resources are equally split among the users, denoted by ψ_j, the task ratio of user j is defined as $\frac{\sum x^*_{j,s}}{\psi_j}$. To study the SI violation by an allocation rule, we restrict the task ratio by 1 (*i.e.*, $\min\left\{\frac{\sum x^*_{j,s}}{\psi_j}, 1\right\}$), and if this value is equal to 1, user j does not benefit from receiving an equal share. We derive the minimum restricted task ratio (*i.e.*, $\min_{j\in\mathcal{J}}\left\{\min\left\{\frac{\sum x^*_{j,s}}{\psi_j}, 1\right\}\right\}$) to find the most disadvantaged user at a given time instance. Note that if an allocation rule satisfies SI, the minimum restricted task ratio equals 1 for any environment at any time. In Fig. 3, the average minimum restricted task ratio and the time ratio that SI is violated during the simulation are plotted versus ρ, when 100 jobs are uniformly sampled from the first 1000 job submission in the Google cluster trace, and the BW capacity is set to 2 MHz. The average minimum restricted task ratio of DRF-$\boldsymbol{\alpha}^{\text{er}}$, TSF-$\boldsymbol{\alpha}^{\text{er}}$, and DRF-ER decreases as ρ increases, *i.e.*, they are further away from satisfying SI when the server heterogeneity increases. Furthermore, DRF-$\boldsymbol{\alpha}^{\text{er}}$, TSF-$\boldsymbol{\alpha}^{\text{er}}$, and DRF-ER violate SI more often when the server heterogeneity increases.

In Fig. 4, we study the performance of the allocation rules in terms of PO. The domination factor denotes how Pareto dominating is an allocation. Given, $x^*_{j,s}$, an allocation rule's number of allocated task to user j on server s at a given time instance, we find the dominating allocation with the maximum social utility (*i.e.*, maximizing $\sum_{j\in\mathcal{J}}\sum_{s\in\mathcal{S}} x_{j,s}$ such that $\sum_{s\in\mathcal{S}} x_{j,s} \geq \sum_{s\in\mathcal{S}} x^*_{j,s}$ for all $j \in \mathcal{J}$). The domination factor at that time instance is defined as the ratio of the social utility of the found dominating allocation to the social utility of the allocation derived from the allocation rule. Note that if an allocation rule satisfies PO, the domination factor at any time instance equals 1 for any environment. In Fig. 4, we illustrate the average domination factor over the simulation time versus ρ, when 100 users are uniformly sampled from the first 1000 job submission in the Google cluster trace, and the BW capacity is set to 2 MHz. We observe that DRF-$\boldsymbol{\alpha}^{\text{er}}$ and TSF-$\boldsymbol{\alpha}^{\text{er}}$ are less Pareto efficient and more often violate PO when the server heterogeneity increases.

Figures 3 and 4 together with Table 2 show that TSF-ER is superior to DRF-$\boldsymbol{\alpha}^{\text{er}}$, TSF-$\boldsymbol{\alpha}^{\text{er}}$, and DRF-ER, both in theory and practice. Therefore, in what follows, we compare the performance

TABLE 3
Average of the users' JCT factor for TSF-ER, MNW, and CRU under fluid task model. The overall efficiency of TSF-ER is at most 6.5% lower than CRU and MNW, which is a price that TSF-ER has to pay to satisfy SP. However, TSF-ER, unlike the other allocation rules, satisfies the four properties altogether.

Allocation rule \ n	**10**	**15**	**20**	15	15	15	15	15	15
ρ	0.7	0.7	0.7	**0.5**	**0.7**	**0.9**	0.7	0.7	0.7
BW	2 MHz	2 MHz	2 MHz	2 MHz	2 MHz	2 MHz	**1 MHz**	**2 MHz**	**3 MHz**
TSF-ER	0.608	0.491	0.410	0.477	0.491	0.500	0.442	0.491	0.516
MNW	0.621	0.510	0.432	0.492	0.510	0.519	0.448	0.510	0.536
CRU	0.623	0.514	0.437	0.495	0.514	0.524	0.448	0.514	0.540

TABLE 4
Average of the users' JCT factor for TSF-ER, MNW, and CRU under the indivisible task model. Unlike MNW and CRU, TSF-ER could be efficiently extended to the indivisible task model and performs well in realistic environments.

Allocation rule \ n	**10**	**15**	**20**	15	15	15	15	15	15	15	15	15
ρ	0.7	0.7	0.7	**0.5**	**0.7**	**0.9**	0.7	0.7	0.7	0.7	0.7	0.7
BW	2 MHz	2 MHz	2 MHz	2 MHz	2 MHz	2 MHz	**1 MHz**	**2 MHz**	**3 MHz**	2 MHz	2 MHz	2 MHz
reallocation slot	256 s	256 s	256 s	256 s	256 s	256 s	256 s	256 s	256 s	**64 s**	**256 s**	**1024 s**
TSF-ER	0.904	0.849	0.797	0.837	0.849	0.853	0.766	0.849	0.877	0.872	0.849	0.845
MNW	0.262	0.174	0.128	0.172	0.174	0.183	0.158	0.174	0.182	0.178	0.174	0.173
CRU	0.305	0.200	0.143	0.219	0.200	0.187	0.186	0.200	0.205	0.212	0.200	0.193

of TSF-ER, *i.e.*, the best egalitarian allocation rule, with MNW and CRU, which represent Nash welfare and utilitarian welfare, respectively.

6.2.1 Small-scale Experiments with Fluid Task Model

In Table 3, we study the impact of the number of arriving users, server heterogeneity (*i.e.*, ρ), and external resource capacity (*i.e.*, BW) on the average of the users' JCT factor under fluid task model. Our simulation results show that CRU and MNW attain better overall efficiency because CRU and MNW maximize the summation and product of users' number of allocated tasks. In particular, by definition, CRU finds the allocation with the best overall efficiency among all the allocations that provide EF and SI. TSF-ER focuses on fairness and equalizes the task shares across the users. Therefore, the overall efficiency of TSF-ER is lower than CRU and MNW, which is a price that TSF-ER has to pay to satisfy SP. However, TSF-ER, unlike the other allocation rules, satisfies the four properties altogether. The maximum drop in the average JCT factor of TSF-ER compared with CRU is only 6.5%, and it occurs when the number of users, ρ, and BW are 20, 0.7, and 2 MHz, respectively.

Table 3 shows that the average JCT factor decreases by increasing the number of arriving users, which makes the competition over the available resources more intensive and reduces the number of tasks, hence the JCT factor that each user can receive. Furthermore, our simulation results show that the performance of the allocation rules does not change by increasing ρ, and they are resilient to server heterogeneity. Decreasing BW makes the external resource the bottleneck resource for more users. Hence, when the BW capacity is too small, our fair multi-resource allocation problem reduces to a fair single-resource allocation problem. Since these allocation rules satisfy EF, they produce the equal BW division in the case of a single-resource environment. Therefore, the allocations they produce become more similar with smaller BW, as confirmed by Table 3.

6.2.2 Small-scale Experiments with Indivisible Task Model

Table 4 illustrates the impact of the number of users, ρ, BW, and the reallocation slot length on the average users' JCT factor under the indivisible task model. It shows that the proposed implementation of TSF-ER for the indivisible task model performs well in realistic environments. Under task indivisibility, this allocation rule successfully provides high utilization (large average JCT factor for all users). Moreover, MNW and CRU fail to adapt to the indivisible task model and perform poorly compared to TSF-ER. This is mainly due to the naive extension of these two allocation rules to the indivisible task model.

Under task indivisibility, TSF-ER attempts to reproduce its fairness and efficiency properties by following the same criterion used in the divisible task model. More specifically, TSF-ER attempts to keep the user's task shares equalized by allocating tasks to the users furthest from their fair share. Under task divisibility, MNW and CRU's criteria for a fair and efficient allocation are maximizing the product of utilities and the sum of utilities, respectively. To extend MNW (rep. CRU) to the indivisible task model so that the product (resp. sum) of utilities is approximately maximized is not straightforward and beyond the scope of this paper. Our simulation result shows that the most obvious extension of MNW and CRU to the indivisible task model, which finds the fractional allocation followed by rounding it, fails to approximate the desired criteria. This result is in line with the observation in [58] that the "integrality gap of the natural fractional relaxation

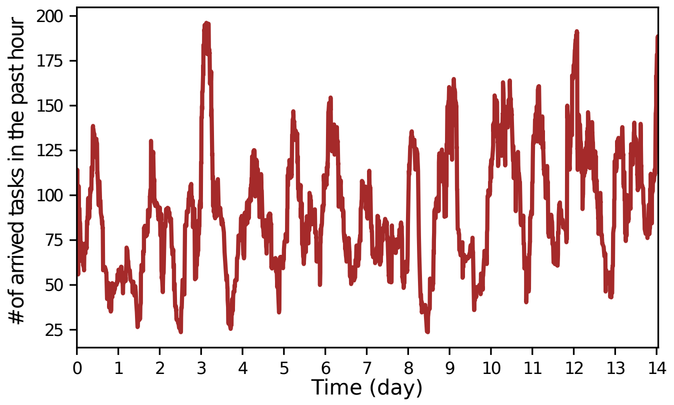


Fig. 5. The number of arrived tasks during the past hour vs. time. We uniformly sampled 4% of the first two weeks of job submissions in the Alibaba cluster trace.

is exponential" for a simpler model with additive utility functions. Unfortunately, we are not aware of any better method to extend MNW and CRU to the indivisible task model.

6.3 Large-scale Experiments with Alibaba Cluster Trace Under Indivisible Task Model

In this section, we evaluate the performance of the allocation rules under the indivisible task model with a large number of users dynamically arriving in a twenty-server system and an external resource. We use the Alibaba cluster trace to simulate an MEC environment for machine learning jobs and synthesize the users' demand for bandwidth (*i.e.*, the external resource) from their CPU demands [54] - [57]. The Alibaba cluster trace describes the AI/ML workloads in the machine-learning-as-a-service provided by the Alibaba platform for artificial intelligence on a GPU cluster with over 1800 servers over a 2-month period. In what follows, we introduce our simulation settings. The notion of JCT factor is defined in Sec. 6.2.

User configuration: Similar to the Google cluster trace, each job in the Alibaba cluster trace is divided into tasks with the same resource demands and the jobs' arrival time, number of submitted tasks, per task execution time, and per task CPU, memory, and GPU demands are included in the trace. Similar to Sec. 6.2, we consider each job in the trace as a distinct user in our simulation setting and synthesize users' required bit-rate based on users' required CPU cores and a random Gamma distribution. We use the same CPU frequency, shape and rate parameter for the Gamma random variable, and wireless channel model as in Sec. 6.2. We uniformly sampled 4% of the first two weeks of job submissions in the Alibaba cluster trace. Figure 5 illustrates the number of arrived tasks in each hour for the sampled job submission.

Server and external resource configuration: We uniformly sample 20 servers from the Alibaba cluster trace. Table 5 demonstrates the server configuration. Moreover, we set the BW of the wireless communication channel to 1.5 GHz.

In the indivisible task model, no allocation rule can satisfy the fairness and efficiency properties (see Sec. 7 for a list of works that satisfy the approximate versions of these properties under the indivisible task model). In what follows, we study the performance of TSF-ER, MNW, and CRU in terms of fairness and efficiency.

Verifying whether an allocation is Pareto-optimal requires solving an integer program and is challenging. Hence, in this

TABLE 5
The server configuration of the large-scale simulation.

CPU cores	Memory	GPUs	Number of servers
64	512 GB	2	9
96	512 GB	2	4
96	512 GB	8	3
96	384 GB	8	4

section, we consider resource utilization as a proxy for the efficiency of an allocation rule. In Figure 6 we illustrate the resource utilization of the allocation rules when the reallocation slot is 1 hour. Note that after 14 days, we assume there is no new job submission to the system. We conjecture that the jittering resource utilization of MNW and CRU is due to their random scheduling policy during the reallocation slots. MNW and CRU perform poorly under the indivisible task model, and their throughput is 35% and 11% worse than that of TSF-ER, respectively. Furthermore, MNW and CRU fail to stabilize the waiting queue, as illustrated in Figure 7. We note that MNW and CRU perform better with smaller reallocation slot length since the reallocation procedure is activated more often. However, due to the severe disruption and high overhead of resource reallocation, in practice we expect the reallocation slot to be at least in the order of hours.

In Figure 8, we consider more realistic values for the reallocation slot and study its impact on the average of the users' JCT factor. For instance, consider the reallocation slot of 2 weeks, which activates the reallocation procedure only once during the scheduling process of the allocation rules. In this case, MNW and CRU randomly assign tasks to the servers almost all the time. However, TSF-ER attempts to equalize the fair share of the users all the time and provides a higher JCT factor level.

Given an allocation rule, let $\mathbf{\Lambda}_j$ be the resource allocation profile of user j derived by that allocation rule. Consider the ratio of the number of tasks that user j can execute with its resource allocation profile to the number of tasks that it can execute with user i's resource allocation profile at a given time instance restricted by 1 (*i.e.*, min $\{u_j(\mathbf{\Lambda}_j)/u_j(\mathbf{\Lambda}_i), 1\}$). The smaller this value is, the more envious user j is toward user i at that time instance. Furthermore, if it equals 1, user j does not envy user i. Hence, we define EF satisfaction (EF-SAT) of user j at that time instance as $\min\left\{\min_{i\in\mathcal{J}}\{u_j(\mathbf{\Lambda}_j)/u_j(\mathbf{\Lambda}_i)\}, 1\right\}$. The minimum EF-SAT (MIN-EF-SAT) of the users at a time instance, *i.e.*, $\min\left\{\min_{i,j\in\mathcal{J}}\{u_j(\mathbf{\Lambda}_j)/u_j(\mathbf{\Lambda}_i)\}, 1\right\}$, represents the EF violation by the most envious user at that time instance. Figure 9 illustrates the average MIN-EF-SAT over the simulation time versus the reallocation slot length, and it shows that TSF-ER substantially outperforms MNW and CRU in terms of envy-freeness.

Similar to Sec. 6.2, we find the task ratio of a user at a time instance as the ratio of the number of tasks allocated to that user to the number of tasks that it can execute when the resources are equally allocated to all users in the system at that time instance. We define SI satisfaction (SI-SAT) of a user as the minimum of its task ratio and 1. Hence, any SI-SAT of less than 1 implies that the user prefers the equal division allocation over what is allocated to it. The minimum SI-SAT (MIN-SI-SAT) of the users at a time instance represents the violation of SI for the most

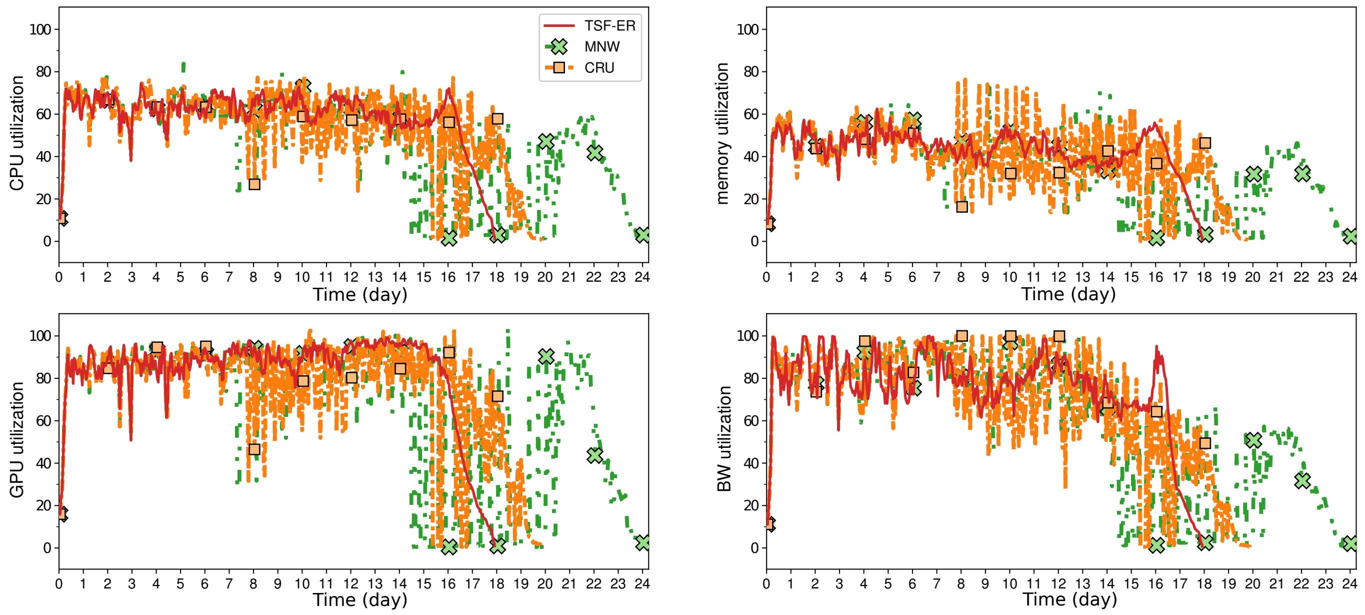


Fig. 6. Averaged resource utilization over the past hour vs. time when the reallocation slot is 1 hour. MNW and CRU perform poorly under the indivisible task model, and their throughput is 35% and 11% worse than that of TSF-ER, respectively.

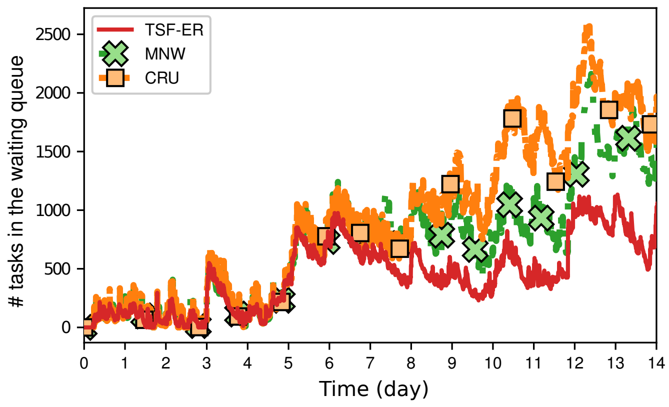


Fig. 7. Number of tasks waiting in the queue vs. time when the reallocation slot is 1 hour. MNW and CRU fail to stabilize the waiting queue.

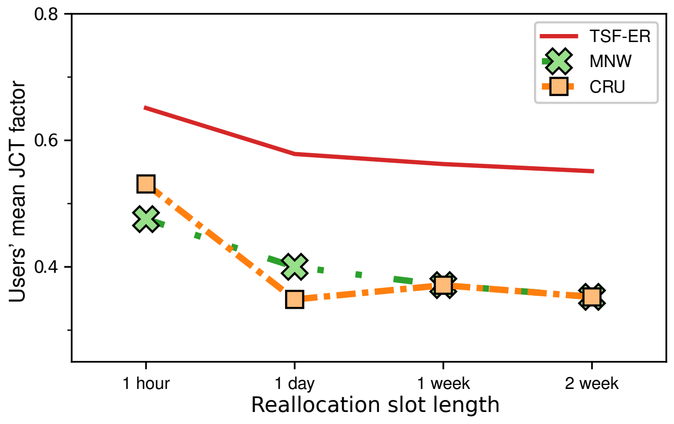


Fig. 8. Users' mean JCT factor vs. reallocation slot. With more realistic values for the reallocation slot, MNW and CRU randomly assign tasks to the servers almost all the time. However, TSF-ER attempts to equalize the fair share of the users all the time and provides a higher JCT factor.

disadvantaged user. Figure 10 illustrates the average MIN-SI-SAT over the simulation time, and it shows that TSF-ER provides far better sharing incentive across the users in comparison to MNW and CRU.

7 Related Work

A fair division of goods is a well-studied problem in microeconomics. There have been two well-known general solutions to this problem, namely Kalai-Smorodinsky (KS) bargaining solution [59] and Nash Bargaining Solution (NBS) [7], [8]. The KS solution equalizes the relative gains (*i.e.* fraction of maximal feasible gains) of all users [31]. When utility functions are linear, KS corresponds to the egalitarian equivalent allocation [47], which finds the maximum ω such that everyone is indifferent between receiving their allocated share or the fraction ω of the total resources [59]. NBS, however, looks for a spot between the utilitarian notion (*i.e.*, maximizing the sum of utilities) and the egalitarian notion (*i.e.*, maximize the minimum utility) by maximizing the utility product [26].

Khamse-ashari *et al.* proposed the notion of dominant virtual share (VDS) [60] to address the trade-off between efficiency and fairness; they chose to allocate resources at each server by applying the so-called α-proportional fairness on VDS (αPF-VDS) [61]. We focus on the case when $\alpha = 1$ since αPF-VDS does not satisfy PO otherwise. αPF-VDS corresponds to NBS when $\alpha = 1$ and finds the same allocation as Competitive Equilibrium from Equal Income (CEEI). In [48], a different environment is studied where a user's demand for a resource may vary in different servers. It was shown in [48] that there exists no allocation rule for this environment such that it can satisfy EF, PO, SP, and SI altogether. Hence, the authors proposed Maximizing Task Product (MTP), which finds an NBS and proved that it satisfies EF, PO, and SI. Unfortunately, CEEI, αPF-VDS, and MTP are not strategy-proof in our environment. Therefore, we rule them out as desirable solutions.

Ghodsi *et al.* studied the problem of multi-resource allocation

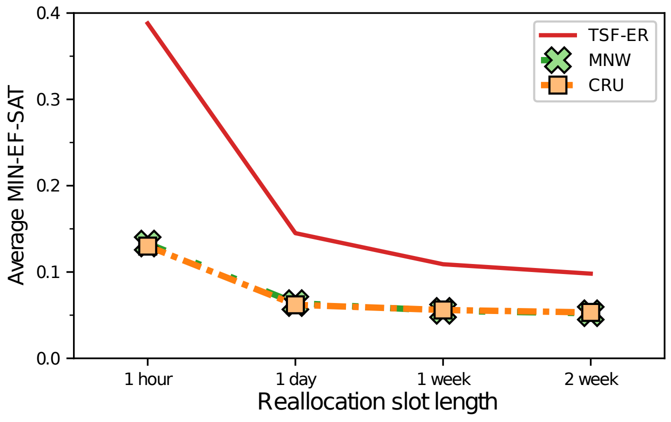


Fig. 9. Average MIN-EF-SAT over the simulation process vs. reallocation slot. By attempting to equalize the fair share of the users, TSF-ER substantially outperforms MNW and CRU in terms of envy-freeness.

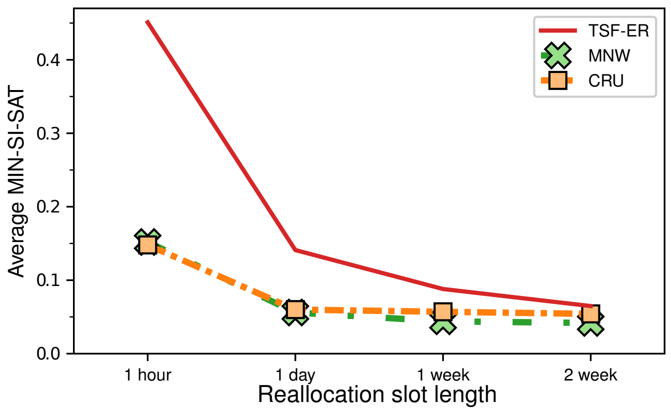


Fig. 10. Average MIN-SI-SAT over the simulation process vs. reallocation slot. By attempting to equalize the fair share of the users, TSF-ER provides far better sharing incentive across the users in comparison with MNW and CRU.

in the context of cloud computing [2], in which users have Leontief preference, *i.e.* each user requires resources in a customized proportion [2], [13], [62], [63]. Furthermore, to avoid waste, no redundant resource should be allocated. Under these circumstances, along with divisible resources, Ghodsi *et al.* proposed DRF, which is an Egalitarian Equivalent (EE) based mechanism. DRF is guaranteed to satisfy PO, EF, SI, and SP altogether [2], and it has been implemented in many practical systems [64], [65], [66]. Subsequently, Li *et al.* studied the egalitarian division under more general preferences (*i.e.*, generalized Leontief preferences) and proposed generalized egalitarian rules that can satisfy the required properties [30]. Parkes *et al.* extended DRF to work under the circumstances that users have zero demands for certain resources [5]. Furthermore, Gutman *et al.* extended DRF to a larger class of user utilities with perfect complement demands [67], where the users' preferences do not allow any substitution between different goods. Moreover, Kash *et al.* extended DRF to a dynamic setting where the users dynamically arrive over time but never depart [13].

DRF has been extended to achieve better fairness-efficiency tradeoff in [68], [69], [70], [71]. Fairness-efficiency tradeoff in multi-resource environment is studied by Dolev *et al.* [63] and Bonald *et al.* [72] in which they proposed other fairness notions based on the system's bottleneck resources. Joe-Wong *et al.* studied the fairness-efficiency tradeoff for single-server multi-resource cloud computing allocation problem by proposing a unifying framework that characterizes the fairness-efficiency tradeoff [62].

The problem of fair resource allocation in cloud computing environments with heterogeneous servers is studied in [3], [6], [33]. In [6], Friedman *et al.* pointed out that DRF can be interpreted as a KS bargaining solution [31]. In this interpretation, DRF applies max-min fairness across the users' normalized task allocation. The normalized task allocation is the number of tasks allocated to a user divided by the maximum number of tasks that user can execute, *i.e.* the number of executable tasks when a user monopolizes the servers. Friedman *et al.* used the KS solution to propose a mechanism for heterogeneous server settings, which also considers the environments where containers are used to achieve performance isolation between tenants [6]. In [3], Wang *et al.* extended DRF to heterogeneous server settings (DRFH) and showed that it satisfies EF, PO, and SP but fails to satisfy SI. To overcome this shortcoming, Wang *et al.* proposed TSF, which finds the KS solution in heterogeneous server settings with divisible resources [33].

In [37], we studied the problem of multi-resource fair allocation in an MEC environment and proposed DRF-ER. It applies max-min fairness across the users' share of dominant resource, and dynamically selects the *best* communication link reservation profile that leads to the highest equalized share of dominant resource for the users. In [37], we showed that DRF-ER could satisfy EF, PO, and SP, but fails to satisfy SI. Unfortunately, extending TSF in the same way to dynamically select the best communication link reservation profile leads into a non-convex optimization problem. Instead, in this work, we present TSF-ER, which redefines the notion of task normalization and finds a KS bargaining solution.

8 Conclusion and Discussion

In this paper, we study the problem of fair resource allocation in systems with heterogeneous servers along with a resource type external to those servers. The external resource is a dedicated resource that exists outside of the servers and is shared by all agents to access the servers. This makes it impossible to satisfy the fairness and efficiency properties with the existing multi-resource fair allocation mechanisms. For this novel environment, we have proposed a multi-resource fair allocation rule termed TSF-ER. We have shown that this allocation rule satisfies important desirable properties. It is envy-free, Pareto optimal, and truthful. Moreover, it satisfies sharing incentive. Trace-driven simulations show that, compared with the two direct extensions of DRF and TSF, namely DRF-$\boldsymbol{\alpha}^{\text{er}}$ and TSF-$\boldsymbol{\alpha}^{\text{er}}$, TSF-ER achieves significant improvements in sharing incentive and Pareto-optimality. Furthermore, unlike DRF-ER, which can be derived by modifying (9f) to equalize the active users' dominant shares and fails to satisfy sharing incentive, TSF-ER satisfies sharing incentive while maintaining effective resource utilization and satisfying Pareto optimality. Moreover, unlike CRU and MNW, which sacrifice strategy-proofness to attain envy-freeness, Pareto optimality, and sharing incentive, TSF-ER maintains strategy-proofness together with the three other properties.

TSF-ER, as presented in Algorithm 1, is suitable for environments with a single external resource. However, in some practical cases, there may exist multiple external resources (*e.g.*, link bandwidth and power). TSF-ER can be extended to such cases by adding more constraints regarding the other external resources to problem (8) in Algorithm 1. Moreover, each user may be assigned a weight, w_i, to indicate its relative importance in the system. Algorithm 1 can be modified to incorporate users with weights by normalizing the number of allocated tasks of a user with respect to $w_i\eta_i$. Note that envy-freeness and sharing incentive should be redefined in the existence of users' weights by scaling a user's allocation in proportion to its weight. The proofs of Theorems 7, 6, 8 and 9 can be easily modified to show that TSF-ER satisfies these properties in the case of multiple external resources and weights.

Finally, TSF-ER can be extended to MEC environments in which different subsets of the servers share some common external resource, *e.g.*, multi-access points MEC environments. If a user's demand for the external resource is fixed, the proofs of Theorems 6, 7, 8 and 9 can be easily modified to show that TSF-ER satisfies all the required properties in such environments. However, in [48], we have proved that if users' demands for the external resource vary at each access point, it is impossible to satisfy the four properties altogether. Another interesting common scenario is where each user is allowed to establish a connection with only one of the MEC access points at any time [73]. To obtain similar results for more general settings remains an open problem for future research.

References

[1] S. Bouveret and J. Lang, "Efficiency and envy-freeness in fair division of indivisible goods: Logical representation and complexity," *Journal of Artificial Intelligence Research*, vol. 32, no. 1, pp. 525–564, June 2008.

[2] 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 Conference on Networked Systems Design and Implementation*, 2011.

[3] W. Wang, B. Liang, and B. Li, "Multi-resource fair allocation in heterogeneous cloud computing systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2822–2835, Oct. 2015.

[4] A. Ghodsi, V. Sekar, M. Zaharia, and I. Stoica, "Multi-resource fair queueing for packet processing," in *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2012.

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

[6] E. Friedman, A. Ghodsi, and C.-A. Psomas, "Strategyproof allocation of discrete jobs on multiple machines," in *Proc. ACM Conference on Economics and Computation*, 2014.

[7] D. K. Foley, "Resource allocation and the public sector," *Yale economic essays*, vol. 7, no. 1, pp. 45–98, 1967.

[8] H. R. Varian, "Equity, envy, and efficiency," *Journal of Economic Theory*, vol. 9, no. 1, pp. 63–91, 1974.

[9] J. Robertson and W. Webb, *Cake-cutting algorithms: Be fair if you can*. CRC Press, 1998.

[10] A. Rubinstein, "Fair division," *Economics and Philosophy*, vol. 13, no. 1, pp. 113–117, 1997.

[11] G. Amanatidis, G. Birmpas, and E. Markakis, "On truthful mechanisms for maximin share allocations," in *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

[12] H. Steihaus, "The problem of fair division," *Econometrica*, vol. 16, no. 1, pp. 101–104, Jan. 1948.

[13] I. Kash, A. D. Procaccia, and N. Shah, "No agent left behind: Dynamic fair division of multiple resources," *Journal of Artificial Intelligence Research*, vol. 51, pp. 579–603, Jan. 2014.

[14] E. Friedman, C.-A. Psomas, and S. Vardi, "Dynamic fair division with minimal disruptions," in *Proc. ACM conference on Economics and Computation*, 2015.

[15] ——, "Controlled dynamic fair division," in *Proc. ACM Conference on Economics and Computation*, 2017.

[16] V. Conitzer, R. Freeman, and N. Shah, "Fair public decision making," in *Proc. ACM Conference on Economics and Computing*, 2017.

[17] B. Fain, K. Munagala, and N. Shah, "Fair allocation of indivisible public goods," in *Proc. ACM Conference on Economics and Computation*, 2018.

[18] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang, "The unreasonable fairness of maximum Nash welfare," *ACM Trans. on Economics and Computation (TEAC)*, vol. 7, no. 3, pp. 1–32, 2019.

[19] G. Amanatidis, E. Markakis, A. Nikzad, and A. Saberi, "Approximation algorithms for computing maximin share allocations," *ACM Trans. on Algorithms (TALG)*, vol. 13, no. 4, pp. 1–28, 2017.

[20] D. Kurokawa, A. D. Procaccia, and J. Wang, "Fair enough: Guaranteeing approximate maximin shares," *Journal of the ACM (JACM)*, vol. 65, no. 2, pp. 1–27, 2018.

[21] J. Garg and S. Taki, "An improved approximation algorithm for maximin shares," *Artificial Intelligence*, 2021.

[22] L. Zhou, "Inefficiency of strategy-proof allocation mechanisms in pure exchange economies," *Social Choice and Welfare*, vol. 8, no. 3, pp. 247–254, 1991.

[23] S. M. Zahedi and B. C. Lee, "Ref: Resource elasticity fairness with sharing incentives for multiprocessors," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 1, pp. 145–160, Feb. 2014.

[24] M. Voorneveld, "From preferences to leontief utility," *Economic Theory Bulletin*, vol. 2, no. 2, pp. 197–204, Oct. 2014.

[25] K. Hashimoto, "Strategy-proofness versus efficiency on the cobb-douglas domain of exchange economies," *Social Choice and Welfare*, vol. 31, no. 3, pp. 457–473, Oct. 2008.

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

[27] A. Muthoo, *Bargaining theory with applications*. Cambridge University Press, 1999.

[28] S. M. Zahedi and B. C. Lee, "Sharing incentives and fair division for multiprocessors," *IEEE Micro*, vol. 35, no. 3, pp. 92–100, 2015.

[29] A. Nicoló, "Efficiency and truthfulness with Leontief preferences. a note on two-agent, two-good economies," *Review of Economic Design*, vol. 8, no. 4, pp. 373–382, Apr. 2004.

[30] J. Li and J. Xue, "Egalitarian division under Leontief preferences," *Economic Theory*, vol. 54, no. 3, pp. 597–622, Nov. 2013.

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

[32] H. Imai, "Individual monotonicity and lexicographic maxmin solution," *Econometrica*, vol. 51, no. 2, pp. 389–401, 1983.

[33] W. Wang, B. Li, B. Liang, and J. Li, "Multi-resource fair sharing for datacenter jobs with placement constraints," in *Proc. ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2016.

[34] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, "Multi-resource packing for cluster schedulers," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 455–466, 2015.

[35] B. Liang, "Mobile edge computing" in *Key Technologies for 5G Wireless Systems*, V. W. S. Wong and R. S. and D. W. K. Ng and L.-C. Wang, Eds. Cambridge University Press, 2017.

[36] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[37] E. Meskar and B. Liang, "Fair multi-resource allocation with external resource for mobile edge computing," in *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2018.

[38] X. Cao, "Preference functions and bargaining solutions," in *IEEE Conference on Decision and Control*, 1982, pp. 164–171.

[39] A. Dhillon, "Extended Pareto rules and relative utilitarianism," *Social Choice and Welfare*, vol. 15, no. 4, pp. 521–542, 1998.

[40] A. Dhillon and J.-F. Mertens, "Relative utilitarianism," *Econometrica*, vol. 67, no. 3, pp. 471–498, 1999.

[41] E. Karni, "Impartiality: definition and representation," *Econometrica*, vol. 66, no. 6, pp. 1405–1415, 1998.

[42] U. Segal, "Let's agree that all dictatorships are equally bad," *Journal of Political Economy*, vol. 108, no. 3, pp. 569–589, 2000.

[43] M. Pivato, "Twofold optimality of the relative utilitarian bargaining solution," *Social Choice and Welfare*, vol. 32, no. 1, pp. 79–92, 2009.

[44] F. Kelly, "Charging and rate control for elastic traffic," *European Trans. on Telecommunications*, vol. 8, no. 1, pp. 33–37, 1997.

[45] A. Hylland and R. Zeckhauser, "The efficient allocation of individuals to positions," *Journal of Political Economy*, vol. 87, no. 2, pp. 293–314, 1979.

[46] D. Bertsimas, V. F. Farias, and N. Trichakis, "The price of fairness," *Operations Research*, vol. 59, no. 1, pp. 17–31, 2011.

[47] H. Moulin, *Fair division and collective welfare*. MIT Press, 2004.

[48] E. Meskar and B. Liang, "Fair multi-resource allocation in mobile edge computing with multiple access points," in *Proc. ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2020, pp. 241–250.

[49] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*. Athena Scientific Belmont, MA, 1997, vol. 6.

[50] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proc. ACM Symposium on Cloud Computing*, 2012.

[51] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding, "MLaaS in the wild: Workload analysis and scheduling in Large-Scale heterogeneous GPU clusters," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Apr. 2022, pp. 945–960.

[52] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang, F. Yang, and L. Zhou, "Gandiva: Introspective cluster scheduling for deep learning," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Carlsbad, CA, Oct. 2018, pp. 595–610.

[53] S. Chaudhary, R. Ramjee, M. Sivathanu, N. Kwatra, and S. Viswanatha, "Balancing efficiency and fairness in heterogeneous GPU clusters for deep learning," in *Proc. European Conference on Computer Systems (EuroSys)*. Association for Computing Machinery, 2020.

[54] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[55] W. Yuan and K. Nahrstedt, "Energy-efficient CPU scheduling for multimedia applications," *ACM Trans. Computer Systems*, vol. 24, no. 3, pp. 292–331, Aug. 2006.

[56] J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with PACE," in *Proc. ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, June 2001.

[57] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time CPU scheduling for mobile multimedia systems," in *Proc. ACM Symposium on Operating Systems Principles*, Oct. 2003.

[58] R. Cole and V. Gkatzelis, "Approximating the Nash social welfare with indivisible items," in *Proc. ACM Symposium on Theory of Computing (STOC)*, 2015, pp. 371–380.

[59] E. A. Pazner and D. Schmeidler, "Egalitarian equivalent allocations: A new concept of economic equity," *The Quarterly Journal of Economics*, vol. 92, no. 4, pp. 671–687, 1978.

[60] J. Khamse-Ashari, I. Lambadaris, G. Kesidis, B. Urgaonkar, and Y. Q. Zhao, "Per-server dominant-share fairness (PS-DSF): A multi-resource fair allocation mechanism for heterogeneous servers," in *Proc. IEEE International Conference on Communications (ICC)*, May 2017.

[61] J. Khamse-Ashari, I. Lambadaris, G. Kesidis, B. Urgaonkar, and Y. Zhao, "An efficient and fair multi-resource allocation mechanism for heterogeneous servers," *IEEE Trans. on Parallel and Distributed Systems*, vol. 29, no. 12, pp. 2686–2699, 2018.

[62] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multiresource allocation: Fairness-efficiency tradeoffs in a unifying framework," *IEEE/ACM Trans. on Networking (TON)*, vol. 21, no. 6, pp. 1785–1798, 2013.

[63] D. Dolev, D. G. Feitelson, J. Y. Halpern, R. Kupferman, and N. Linial, "No justified complaints: On fair sharing of multiple resources," in *Proc. ACM Conference on Innovations in Theoretical Computer Science (ITCS)*, 2012.

[64] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in *Proc. USENIX Conference on Networked Systems Design and Implementation*, 2011.

[65] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler, "Apache Hadoop YARN: Yet another resource negotiator," in *Proc. ACM Annual Symposium on Cloud Computing*, 2013.

[66] A. Beltre, P. Saha, and M. Govindaraju, "KubeSphere: An approach to multi-tenant fair scheduling for Kubernetes clusters," in *IEEE Cloud Summit*, 2019, pp. 14–20.

[67] A. Gutman and N. Nisan, "Fair allocation without trade," in *Proc. International Conference on Autonomous Agents and Multiagent Systems*, 2012.

[68] H. Hamzeh, S. Meacham, B. Virginas, K. Khan, and K. Phalp, "MLF-DRS: A multi-level fair resource allocation algorithm in heterogeneous cloud computing systems," in *Proc. IEEE International Conference on Computer and Communication Systems (ICCCS)*, 2019.

[69] Youngmi Jin and M. Hayashi, "Efficiency comparison between proportional fairness and dominant resource fairness with two different type resources," in *Proc. Annual Conference on Information Science and Systems (CISS)*, 2016.

[70] L. Zhao, M. Du, and L. Chen, "A new multi-resource allocation mechanism: A tradeoff between fairness and efficiency in cloud computing," *China Communications*, vol. 15, no. 3, pp. 57–77, 2018.

[71] Y. Jin and M. Hayashi, "Trade-off between fairness and efficiency in dominant alpha-fairness family," in *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018.

[72] T. Bonald and J. Roberts, "Multi-resource fairness: Objectives, algorithms and performance," in *Proc. ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2015.

[73] E. Meskar and B. Liang, "MAGIKS: Fair multi-resource allocation game induced by kalai-smorodinsky bargaining solution," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 797–810, 2022.

9 Appendix

In this section, we present the proofs for the theorems in Sec. 3 and the lemmas in Sec. 4.

9.1 Proof of Theorem 1

Proof. Due to restricting CRU to the set of feasible envy-free allocations with sharing incentive, CRU trivially satisfies EF and SI. To prove Pareto optimality of the CRU allocations, assume by way of contradiction that CRU does not satisfy PO. Thus, there exists an environment E such that its CRU allocation is not on the Pareto frontier of $\mathcal{X}(E)$. Let $\mathbf{x}$ denote the CRU allocation for this environment. Hence, there exists some $\mathbf{y} \in \mathcal{X}(E)$ such that $\sum_{s\in\mathcal{S}} y_{i,s} \geq \sum_{s\in\mathcal{S}} x_{i,s}$ for all $i \in \mathcal{J}$ and $\sum_{s\in\mathcal{S}} y_{j,s} > \sum_{s\in\mathcal{S}} x_{j,s}$ for some $j \in \mathcal{J}$. We build upon $\mathbf{y}$ to create another allocation $\mathbf{z}$ such that $\sum_{s\in\mathcal{S}} z_{i,s} = \sum_{s\in\mathcal{S}} x_{i,s}$ for any user $i \in \mathcal{J}$. This can be achieved by setting $z_{i,s} = \frac{\sum_{s\in\mathcal{S}} x_{i,s}}{\sum_{s\in\mathcal{S}} y_{i,s}} y_{i,s}$ for any user $i \in \mathcal{J}$ and server $s \in \mathcal{S}$. Since $\sum_{s\in\mathcal{S}} y_{j,s} > \sum_{s\in\mathcal{S}} x_{j,s}$ for some $j \in \mathcal{J}$, there exists some server $t \in \mathcal{S}$ such that all resources are under-utilized on server t. We construct $\mathbf{z}$ by equally splitting the unutilized resources of t among all users and adding it to the allocation $\mathbf{y}$. It is easy to check that $\mathbf{z} \in \mathcal{X}(E)$. Moreover, since $\mathbf{x} \in \mathcal{X}_{\text{SI}}(E)$ and $\sum_{s\in\mathcal{S}} z_{i,s} \geq \sum_{s\in\mathcal{S}} x_{i,s}$ for all $i \in \mathcal{J}$, we have $\mathbf{z} \in \mathcal{X}_{\text{SI}}(E)$. Finally, we prove that no user prefers another user's allocation, *i.e.*, $\mathbf{z} \in \mathcal{X}_{\text{EF}}(E)$. This contradicts $\mathbf{x}$ being the CRU allocation for this environment, since $\mathbf{z} \in \mathcal{X}(E) \cap \mathcal{X}_{\text{EF}}(E) \cap \mathcal{X}_{\text{SI}}(E)$ and $\sum_{j\in\mathcal{J}} \frac{\sum_{s\in\mathcal{S}} z_{j,s}}{\eta_j} > \sum_{j\in\mathcal{J}} \frac{\sum_{s\in\mathcal{S}} x_{j,s}}{\eta_j}$.

We show that CRU fails to satisfy SP by presenting a counterexample. Consider the simple environment with a single server with a single resource, *e.g.*, CPU, and BW as the external resource. Let there be 10 units of CPU and 10 units of BW available in the environment. There are two users in the system. User A requires 6 units of CPU and 3 units of BW per task, and user B requires 2 units of CPU and 9 units of BW per task. CRU allocates 0.625 task to user B. However, user B can benefit by reporting the fake demand of 4 units of CPU and 7 units of BW per task. With this fake demand, CRU allocates 0.832 tasks to user B. Hence user B can improve it number of tasks from 0.625 to $0.832 \times \min\{4/2, 7/9\} \approx 0.647$. Therefore, CRU does not satisfy SP. □

9.2 Proof of Theorem 2

Proof. Instead of the geometric mean of the users' utilities, we can maximize its logarithm, *i.e.*,

$$\begin{aligned}
\max_{\mathbf{x}} \quad & \sum_{j\in\mathcal{J}} \log \sum_{s\in\mathcal{S}} x_{j,s} && \text{(12a)}\\
\text{s.t.} \quad & \sum_{j\in\mathcal{J}} x_{j,s}\, d_{j,r} \le c_{s,r}, \forall r\in\mathcal{R},\, s\in\mathcal{S}, && \text{(12b)}\\
& \sum_{j\in\mathcal{J}} x_{j,s}\, d_{j,\text{er}} \le \alpha_s^{\text{er}}, \forall s\in\mathcal{S}, && \text{(12c)}\\
& \sum_{j\in\mathcal{J}} x_{j,s} \ge 0, \forall j\in\mathcal{J},\, s\in\mathcal{S}. && \text{(12d)}
\end{aligned}$$

Pareto optimality of MNW is trivial. We show that MNW fails to satisfy SP by presenting a counter example. Let $\mathbf{x}^*$, $p^*_{s,r}$ and p^*_{er} denote the primal and dual solution of Problem (12). The allocation-price pair $(\mathbf{x}^*, (p^*_{s,r}, p^*_{\text{er}}))$ forms the CEEI of the virtual market constructed by the same set of users and resources and assigning an artificial price to each resource[3]. Note that at CEEI, each user receives its optimal utility that can be afforded with its artificial \$1 income, and the price of each user's received resource bundle is exactly \$1. This directly implies that MNW satisfies EF. Furthermore, the KKT conditions of Problem (12) imply that with $(p^*_{s,r}, p^*_{\text{er}})$, the total price for purchasing all resources in the system is $|\mathcal{J}|$. Hence, the price for purchasing the equal division share (*i.e.*, $1/|\mathcal{J}|$ of each server) is \$1. Therefore, no user prefers the equal division share, since with $\mathbf{x}^*$ and $(p^*_{s,r}, p^*_{\text{er}})$, no user prefers any bundle of resource that costs \$1 to its allocated resource bundle. Thus, MNW satisfies SI.

We use the exact same counterexample presented in the proof of Theorem 1 to prove that MNW does not satisfy SP. MNW allocates 0.625 task to user B with the true reported demand. However, MNW allocates 1 task to user B when it reports the fake demand of 4 units of CPU and 7 units of BW per task. Hence user B can improve its number of tasks from 0.625 to $1 \times \min\{4/2, 7/9\} \approx 0.777$. Therefore, MNW does not satisfy SP. □

9.3 Proof of Theorem 3

Proof. Consider a server configuration in which the servers are not the scaled version of the exact same server, *i.e.*, there exist $s_1, s_2 \in \mathcal{S}$ and $r_1, r_2 \in \mathcal{R}$ such that $\frac{c_{s_1,r_1}}{c_{s_1,r_2}} \neq \frac{c_{s_2,r_1}}{c_{s_2,r_2}}$. Note that for any $s\in\mathcal{S}$ and $r\in\mathcal{R}$, $\sum_{s\in\mathcal{S}} c_{s,r} = c^{\text{er}} = 1$. For such an environment, given any feasible external resource reservation profile $\boldsymbol{\alpha}^{\text{er}}$, there exists some $r_0 \in \{r_1, r_2\}$, and $\delta > 0$ such that $\frac{\alpha^{\text{er}}_{s_2}}{c_{s_2,r_0}} < \delta < \frac{\alpha^{\text{er}}_{s_1}}{c_{s_1,r_0}}$. We construct a simple scenario in which there exists only one user in the environment, to show that DRF-$\boldsymbol{\alpha}^{\text{er}}$ does not satisfy PO. Let us denote this user by j. We construct the demand profile of user j such that $\delta d_{j,r_0} = d_{j,\text{er}}$, and for any $r \in \mathcal{R}/\{r_0\}$ and $s\in\mathcal{S}$, $\frac{c_{s,r}}{d_{j,r}} > \max\left\{\frac{c_{s,r_0}}{d_{j,r_0}}, \frac{\alpha^{\text{er}}_s}{d_{j,\text{er}}}\right\}$. With the external resource reservation profile $\boldsymbol{\alpha}^{\text{er}}$, the number of allocated tasks by DRF-$\boldsymbol{\alpha}^{\text{er}}$ is

$$\begin{aligned}
&\sum_{s\in\mathcal{S}} \min\left\{\frac{\alpha^{\text{er}}_s}{d_{j,\text{er}}}, \frac{c_{s,r_0}}{d_{j,r_0}}\right\}\\
&= \min\left\{\frac{\alpha^{\text{er}}_{s_1}}{d_{j,\text{er}}}, \frac{c_{s_1,r_0}}{d_{j,r_0}}\right\} + \min\left\{\frac{\alpha^{\text{er}}_{s_2}}{d_{j,\text{er}}}, \frac{c_{s_2,r_0}}{d_{j,r_0}}\right\}\\
&+ \sum_{s\neq s_1,s_2} \min\left\{\frac{\alpha^{\text{er}}_s}{d_{j,\text{er}}}, \frac{c_{s,r_0}}{d_{j,r_0}}\right\}\\
&< \min\left\{\frac{1}{d_{j,\text{er}}}, \frac{1}{d_{j,r_0}}\right\}.
\end{aligned}$$

However, without using any external resource reservation profile, we can allocate all the servers to user j and end up $\min\left\{\frac{1}{d_{j,r_0}}, \frac{1}{d_{j,\text{er}}}\right\}$ tasks. Hence, DRF-$\boldsymbol{\alpha}^{\text{er}}$ does not satisfy PO. Furthermore, since user j is the only user in the system, the per server equal share is equivalent to allocating the entire system to this user. Hence, similar to the case for PO, we can show that $\boldsymbol{\alpha}^{\text{er}}$-DRF does not satisfy SI. We can generalize this result to multi-user scenarios by adding more users with the exact same demand profiles. □

9.4 Proof of Lemma 1

Proof. If $i \in \mathcal{J}_{\text{active}}(E;\lambda)$, the procedure LP-TaskShare equalizes the task share of user i and j and the inequality in Lemma 1 holds. If $i \notin \mathcal{J}_{\text{active}}(E;\lambda)$, it means that at some iteration n of the while loop of procedure TSF-ER, user i was saturated and its task share was not increased afterwards. The task share of user j at iteration n is not less than user i's task share and its task share does not decrease in the remaining iterations. □

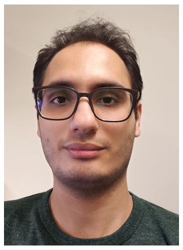
9.5 Proof of Lemma 3

Proof. Since the normalized number of tasks allocated to user j is not increased in environment E', we have $\frac{\sum_{s\in\mathcal{S}} \lambda_{j,s}(E')}{\eta'_j} \le \frac{\sum_{s\in\mathcal{S}} \lambda_{j,s}(E)}{\eta_j}$. Therefore, for any $r \in \hat{\mathcal{R}}$, $\sum_{s\in\mathcal{S}} \lambda_{j,s}(E') \frac{d'_{j,r}}{d_{j,r}} \le \sum_{s\in\mathcal{S}} \lambda_{j,s}(E) \frac{\eta'_j}{\eta_j} \frac{d'_{j,r}}{d_{j,r}}$. Hence, $\sum_{s\in\mathcal{S}} \lambda_{j,s}(E') \min_{r\in\hat{\mathcal{R}}} \left\{\frac{d'_{j,r}}{d_{j,r}}\right\} \le \sum_{s\in\mathcal{S}} \lambda_{j,s}(E) \frac{\eta'_j}{\eta_j} \min_{r\in\hat{\mathcal{R}}} \left\{\frac{d'_{j,r}}{d_{j,r}}\right\}$. By Lemma 2, $u_j(\Lambda_j(E')) = \sum_{s\in\mathcal{S}} \lambda_{j,s}(E') \min_{r\in\hat{\mathcal{R}}} \left\{\frac{d'_{j,r}}{d_{j,r}}\right\}$. Therefore, $u_j(\Lambda_j(E')) \le \sum_{s\in\mathcal{S}} \lambda_{j,s}(E) \min_{r\in\hat{\mathcal{R}}} \left\{\frac{\eta'_j d'_{j,r}}{\eta_j d_{j,r}}\right\}$. Furthermore, Lemma 1 implies that $\min_{r\in\hat{\mathcal{R}}} \left\{\frac{\eta'_j d'_{j,r}}{\eta_j d_{j,r}}\right\} \le 1$. Therefore, $u_j(\Lambda_j(E')) \le u_j(\Lambda_j(E))$. □

9.6 Proof of Lemma 4

Proof. It suffices to prove that user k's normalized number of tasks is not decreased, *i.e.*, $\sum_{s\in\mathcal{S}} \lambda_{k,s}(E')/\eta'_k \ge \sum_{s\in\mathcal{S}} \lambda_{k,s}(E)/\eta_k$. Note that $\eta'_k = \eta_k$ for any $k \neq j$. Assume by way of contradiction that user k's normalized number of tasks is decrease. We can decrease the number of tasks allocated to user j (*i.e.*, $\sum_{s\in\mathcal{S}} \lambda_{j,s}(E')$) and increase the number of tasks allocated to user k (*i.e.*, $\sum_{s\in\mathcal{S}} \lambda_{k,s}(E')$). This contradicts the premise of max-min fairness of TSF-ER allocation across users' normalized number of allocated tasks. □

3. This can be confirmed by examining the KKT conditions of problem (12).



Erfan Meskar received B.Sc. degree in electrical engineering from the Amirkabir University of Technology, Tehran, Iran, in 2013, the M.A.Sc. degree in electrical and computer engineering from McMaster University, Hamilton, Canada, in 2015, and the Ph.D. degree in electrical and computer engineering from University of Toronto, Toronto, Canada, in 2022. He is currently a postdoctoral researcher at University of Toronto and his current research interests are in algorithmic fairness, algorithmic game theory, mobile edge computing, and network anomaly detection.

Ben Liang (Fellow, IEEE) received honors-simultaneous B.Sc. (valedictorian) and M.Sc. degrees in Electrical Engineering from Polytechnic University (now the engineering school of New York University) in 1997 and the Ph.D. degree in Electrical Engineering with a minor in Computer Science from Cornell University in 2001. He was a visiting lecturer and post-doctoral research associate at Cornell University in the 2001 - 2002 academic year. He joined the Department of Electrical and Computer Engineering at the University of Toronto in 2002, where he is now Professor and L. Lau Chair in Electrical and Computer Engineering. His current research interests are in networked systems and mobile communications. He is an associate editor for the IEEE Transactions on Mobile Computing and has served on the editorial boards of the IEEE Transactions on Communications, the IEEE Transactions on Wireless Communications, and the Wiley Security and Communication Networks. He regularly serves on the organizational and technical committees of a number of conferences. He is a Fellow of IEEE and a member of ACM and Tau Beta Pi.