MAGIKS: Fair Multi-resource Allocation Game Induced by Kalai-Smorodinsky Bargaining Solution

Erfan Meskar and Ben Liang, *Fellow, IEEE*

We study the problem of fair and efficient mechanism design for allocating multiple resources in multiple servers among a set of users with Leontief utilities. This problem is motivated by a mobile edge computing environment where each mobile user cannot establish a wireless connection simultaneously to multiple edge servers. Each user is a selfish utility maximizing agent that chooses a single server for its job execution. When a server is shared by multiple users, a resource allocation rule decides the utility that each user must receive. Our goal is to design a mechanism that always admits a Nash Equilibrium (NE), *i.e.*, a state where no user has incentive to change its server, that (1) can be reached in polynomial time and (2) provides fair and efficient resource allocation. We propose the Multi-resource Allocation Game Induced by Kalai-Smorodinsky bargaining solution (MAGIKS) and prove that under discrete resource demands it finds an NE in $\mathcal{O}\left(\text{poly}(n)\right)$ moves for any fixed server configuration, where n is the number of users. Furthermore, MAGIKS satisfies envy-freeness, sharing incentive, and Pareto optimality on each server. Regarding fairness among users in different servers, we prove that MAGIKS satisfies 2-approximate envy-freeness and maximin share guarantee. Moreover, we show that 2-approximate envy-freeness is the best that any mechanism that satisfies local Pareto optimality can achieve at its NEs.

***Index Terms*—Envy-Freeness, Fair Allocation, Maximin Share Guarantee, Nash Equilibrium**

I. Introduction

THERE is a long history of mathematically rigorous studies on fair division of resources among a set of agents [1]–[6]. Developing a fair division mechanism is of immense significance to guaranteeing the quality of experience for different agents. This problem has been considered from different perspectives in economics and computer science and arises in various real-world settings such as auctions, airport traffic management, spatial resource allocation, fair scheduling, and allocation of tasks to workers.

Defining fairness is a critical point in resource allocation mechanisms. When a single resource is shared among identical agents, the fairest allocation is to divide the resource equally. In more complicated settings, however, it is challenging to define fairness. For instance, imagine sharing among a group of n agents a heterogeneous cake, *i.e.*, two pieces of the cake may differ in terms of their toppings. When the agents have different preferences over the pieces, cutting the cake into n pieces of equal size could be far from fair.

The preferences of an agent are expressed by a utility function. In this paper, we focus on the Leontief utility function, which is one of the most widely used utility functions in economic modelling. It describes an agent's preferences on non-interchangeable resources in a fixed proportion for deriving a positive utility [7]–[10]. For example, in the context of cloud computing, a server's resources such as CPU and memory are non-interchangeable for task execution, and the number of tasks that a user can execute is expressed by a Leontief function.

We consider the problem of fair resource allocation with m servers, each containing a set of l complementary divisible resources, among n users. Each user prefers the allocation that provides the highest number of tasks. For instance, consider a user that requires 1 CPU core and 2GB of memory to execute one task. By receiving an allocation of 2 CPU cores and 5GB of memory, this user can execute $\min\{2/1, 5/2\} = 2$ tasks since the resources are complementary. Hence, the number of tasks that a user can execute can be represented by a Leontief utility function of the form $\min\{x_1/\alpha_1, \ldots, x_l/\alpha_l\}$. Similar settings have been studied by economists and computer scientists [7]–[16]. However, there are two features unique to our setting. First, the users in our system cannot utilize more than one server at the same time. For instance, consider a mobile edge computing environment in which the mobile users cannot establish a wireless connection to multiple edge nodes. Consequently, each user must be associated with a server and share it with other users with the same association, under a prespecified allocation rule local to each server. Second, each user is selfish and chooses the server that leads to the best resource allocation for itself. Hence, the local resource allocation rule induces a game among the users, and each user prefers to be associated with a server that provides it with the maximum utility given the other users' server association.

Our goal is to design a fair and efficient mechanism for this game. In the context of fair division of goods, fairness and efficiency of an allocation are evaluated by observing whether it satisfies several core properties, namely envy-freeness, sharing incentive, and Pareto optimality [7]–[16]. Envy-freeness ensures that no agent prefers the allocation of another agent. Sharing incentive provides performance isolation, as it guarantees a minimum utilization for each user irrespective of the demands of the other users, *i.e.*, it guarantees that each user's outcome is no worse than $1/n$ of the outcome in the hypothetical scenario where it receives all the resources. Pareto optimality embodies efficient resource utilization. With Pareto optimality, it is impossible to increase the outcome of a user without decreasing that of another user. These three criteria can be applied directly to each server. We denote them by Local Envy-Freeness (LEF), Local Sharing

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

E. Meskar and B. Liang are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada (e-mail: {emeskar, liang}@ece.utoronto.ca).

incentive (LSI), and Local Pareto optimality (LPO).

Furthermore, the mechanism must ensure that a Nash Equilibrium (NE) always exists, *i.e.*, a state of association such that no user has any incentive to change its server. Moreover, all possible NE associations derived by the mechanism must guarantee fairness among users across different servers. We consider α-approximate Envy-Freeness at NE (αEF-NE) and MaxiMin Share guarantee at NE (MMS-NE). αEF-NE ensures that at any NE association, the users approximately prefer their allocation to any other user's allocation on any other server. MMS-NE ensures that at any NE association, no user's utility is worse than its utility under the well-known cut-and-choose protocol.

The main contributions of this paper are as follows:

- We propose the Multi-resource Allocation Game Induced by the Kalai-Smorodinsky bargaining solution (MAGIKS). MAGIKS uses the Kalai-Smorodinsky (KS) bargaining solution as its local allocation rule. Starting from any initial user-server association, MAGIKS randomly picks a user that prefers to change its associated server in each step, and moves that user accordingly.
- MAGIKS can be considered as a special case of the vector scheduling game in which the maximum demand of each user is equal to 1. The vector scheduling game is known to have NEs, but there is no existing analysis to confirm whether an NE for this game can be reached in polynomial time. In contrast, we prove that under discrete resource demands, MAGIKS terminates after $\mathcal{O}(\text{poly}(n))$ moves for any fixed server configuration.
- We show that MAGIKS satisfies LEF, LSI, and LPO, among users who are associated with the same server. To show fairness of MAGIKS among users who have chosen different servers, we prove that it satisfies 2EF-NE and MMS-NE. Moreover, we show that 2EF-NE is the best that any mechanism that satisfies LPO can achieve at its NEs.
- Our simulation results further suggest that the average number of moves for MAGIKS to convergence increases only linearly with the number of users and servers. Moreover, MAGIKS performs well also in terms of the utilitarian social objective.

The organization of this paper is as follows. In Section II, we summarize the existing fair allocation rules related to our problem. In Section III, we describe the system model. In Section IV, we describe the mechanism properties and discuss the existing challenges in designing a fair and efficient mechanism. In Section V, we present MAGIKS and analyze its NE, fairness, and efficiency properties. In Section VI, we further evaluate the performance of MAGIKS via simulation, followed by concluding remarks in Section VII.

II. Related work

The problem of fair allocation of resources or goods has been studied under various types of utility functions, including Leontief utilities [7]–[16], additive utilities [17]–[21], and Cobb-Douglas utilities [22]–[24]. In this section, we focus on the works on fair allocation for Leontief utilities. Furthermore, we review the works on the vector scheduling game, which is closely related to MAGIKS.

A. Fair Multi-resource Allocation Rules

Ghodsi *et al.* studied the problem of fair multi-resource allocation in a cloud computing server with l resources and n users with Leontief utilities. They proposed Dominant Resource Fairness (DRF), a non-wasteful resource allocation policy, which can satisfy envy-freeness, sharing incentive, and Pareto optimality, among other properties [7]. DRF can be interpreted as a KS allocation rule [25], [26]. It finds the lexicographic maxmin solution after a certain normalization of utilities.

Subsequent to DRF, researchers have extended the KS allocation rule to other problem settings. Parkes *et al.* modified DRF to circumvent possible zero demands for the resources [8]. Ghodsi *et al.* extended dominant resource fairness to fair queueing for packet processing in middleboxes [9]. Li *et al.* studied the KS allocation rule under more general preferences (*i.e.*, generalized Leontief preferences) and proposed generalized egalitarian rules that can satisfy the required properties [10]. Wang *et al.* proposed DRFH, which extends the dominant resource fairness to the setting with multiple heterogeneous servers [11]. For the same setting with additional accommodations for task placement constraints, Wang *et al.* proved that the KS allocation rule satisfies all the desirable fairness and efficiency properties [12]. Meskar *et al.* proposed DRF-ER, which extends DRF to mobile edge computing environments [13].

The Nash Bargaining Solution (NBS) is another well-known fair allocation rule, which maximizes the product of the users' utility [27], [28]. Meskar *et al.* used NBS to fairly allocate resources among mobile edge computing users that have multiple Leontief utility functions for each access point [14]. They proved that NBS satisfies envy-freeness, sharing incentive, and Pareto optimality for such settings. Khamse-Ashari *et al.* proposed the notion of dominant virtual share (VDS) [15] 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) [16]. They presented their allocation rule by formulating a game in which the servers are the players. A server's strategy is the number of tasks that it allocates to each user. In the case where $\alpha = 1$, which is the only case that αPF-VDS satisfies Pareto optimality, αPF-VDS corresponds to the NBS allocation rule.

What distinguishes our fair allocation problem from those in [7]–[16] is that the users are the decision-makers and choose which server they want. Our job is to design a mechanism that allocates resources to the users based on their chosen server association and their resource demand.

B. The Vector Scheduling Game

From the game-theoretical perspective, the works in [29], [30] are closely related to our problem. In [29], Ye *et al.* proposed the vector scheduling game for multi-dimensional load balancing. They proved that this game always admits an NE and proposed upper and lower bound for the price of

anarchy. In [30], Epstein *et al.* improved the results in [29] and proved the existence of an NE for more general settings, where a user's job vector can have entries with the value 0. Furthermore, they improved the analysis of the price of anarchy and presented tight bounds for it. The results in [29], [30] can be extended to our game to prove the existence of NEs. However, [29] and [30] does not provide analysis on the time complexity to reach their NEs. While such time complexity has been extensively studied for single-dimension load balancing games [31], there is no known results for the vector scheduling game. Moreover, [29] and [30] did not study the fairness properties of the outcome of the game.

III. System Model

Let $\mathbb{N}$ and $\mathbb{R}_{++}$ denote the set of natural numbers and the set of positive real numbers, respectively. For $k \in \mathbb{N}$, we define $[k] := \{1, ..., k\}$.

A. Servers and Users

We consider a set of m servers each equipped with l divisible resources. The servers are considered to be homogeneous and each server has the capacity vector $\mathbf{c} = (c_1, \ldots, c_l)$, where $c_r > 0$ denotes the capacity of resource r on each server.

Let n denote the number of users. Each user chooses a single server for its job execution, *e.g.*, a mobile edge computing environment in which mobile users cannot establish a wireless connection to multiple edge nodes. An example of such a system is illustrated in Figure 1. We say that user j is associated with server s if it chooses that server. Furthermore, the users are selfish utility maximizers. Hence, each user prefers to be associated with a server that gives it a higher utility given the associations of the other users.

Each user requires resources in a customized proportion, *i.e.*, the users have Leontief utilities. For example, the resources in an edge computing or cloud computing environment are non-interchangeable and the number of tasks that a user can execute can be expressed as a Leontief function of the form $\min\{x_1/\alpha_1, \ldots, x_l/\alpha_l\}$, where $x_1, \ldots, x_l$ denote the input of the function and $\alpha_1, \ldots, \alpha_l$ are positive constants. Let $u_j(.)$ denote the utility function of user j. The users report their utility functions by submitting their demands for each resource, *i.e.*, $d_{j,r}$, per unit of utility/task. We denote the demand vector of user j by $\mathbf{d}_j = (d_{j,1}, \ldots, d_{j,l})$ and say that the demand vector of user j is positive if $d_{j,r} > 0$ for all $r \in [l]$. Let s be the server that user j is associated with and $\mathbf{b}_{j,s} = (b_{j,s,1}, \ldots, b_{j,s,l})$ be the vector of allocated resources on server s to user j. Then, the utility of user j with this allocation will be $u_j(\mathbf{b}_{j,s}) = \min_{r \in [l]} \left\{ \frac{b_{j,s,r}}{d_{j,r}} \right\}$.

B. Local Allocation Rule

A local allocation rule allocates the resources of a server to its associated users, constrained by its resource capacity. We restrict ourselves to non-wasteful allocation rules, *i.e.*, all resources allocated to any user must be completely utilized. Hence, allocated resources to a user should follow the same proportion as the user's submitted demands, *i.e.*, the amount of any resource $r \in [l]$ that is allocated to user j is equal to $\alpha d_{j,r}$ for some utility $\alpha \geq 0$. Therefore, for any non-wasteful allocation rule, allocating resources is equivalent to allocating utilities. Let λ be a local allocation rule. Given $A_s \subseteq [n]$, the set of users associated with server s, and any user $j \in A_s$, we denote the utility that λ allocates to user j by $\lambda_j(A_s)$. The local allocation rules should satisfy the following capacity constraint.

$$\sum_{j \in A_s} \lambda_j(A_s)\, d_{j,r} \leq c_r, \ \forall r \in [l]. \tag{1}$$

C. The Multi-resource Allocation Game

By designing the local allocation rule, a game will be induced among the users with the servers as their strategy set. Let ρ_j denote user j's associated server. Then $A_s = \{j \mid \rho_j = s\}$. User j is *unhappy* if it prefers to change its association to a different server, *i.e.*, user j is unhappy if for some $s \in [m]$,

$$\lambda_j\left(A_{\rho_j}\right) < \lambda_j\left(A_s \cup \{j\}\right). \tag{2}$$

Let $\Pi_m([n])$ denote the set of all m-partitions of the set of users $[n]$. An association $\mathcal{A}^* = (A_1^*, \ldots, A_m^*) \in \Pi_m([n])$ is a Nash Equilibrium (NE) if all users are happy with their association, *i.e.*, for all $j \in [n]$ and $s \in [m]$, $\lambda_j(A^*_{\rho_j^*}) \geq \lambda_j(A_s^* \cup \{j\})$, where ρ_j^* is the server with which user j is associated. We refer to such an association as an NE association.

Our goal is to design a multi-resource allocation game that always admits an NE association. Moreover, the NE association of the designed game should be reached in polynomial time and it should provide fair and efficient resource allocation.

IV. Preliminaries: Fairness Properties and Challenges

We build upon the existing properties in the fair allocation literature to define fairness and efficiency in our problem.

A. Local Fairness Properties

In [7]–[16], the notions of envy-freeness, sharing incentive, and Pareto optimality have been used to evaluate the performance of an allocation rule. We require the designed mechanism to satisfy these properties locally, *i.e.*, for each server.

Definition 1 (Local Envy-Freeness (LEF))**.** *A mechanism with local allocation rule λ satisfies LEF, if for any $n, m, l \in \mathbb{N}$, any $\mathbf{c} \in \mathbb{R}_{++}^l$, any non-empty $A_s \subseteq [n]$, and any demand profile $\{\mathbf{d}_j\}_{j \in A_s} \in \mathbb{R}_{++}^{n \times l}$, there exists no user $i \in A_s$ that prefers the allocation of another user $j \in A_s$, i.e.,*

$$\lambda_i(A_s) \geq u_i\left(\mathbf{b}_{j,s}\right) = \min_{r \in [l]} \left\{ \frac{\lambda_j(A_s) d_{j,r}}{d_{i,r}} \right\},$$

where $\mathbf{b}_{j,s} = (\lambda_j(A_s) d_{j,1}, \ldots, \lambda_j(A_s) d_{j,l})$ is the resources vector that is allocated to user j on server s.

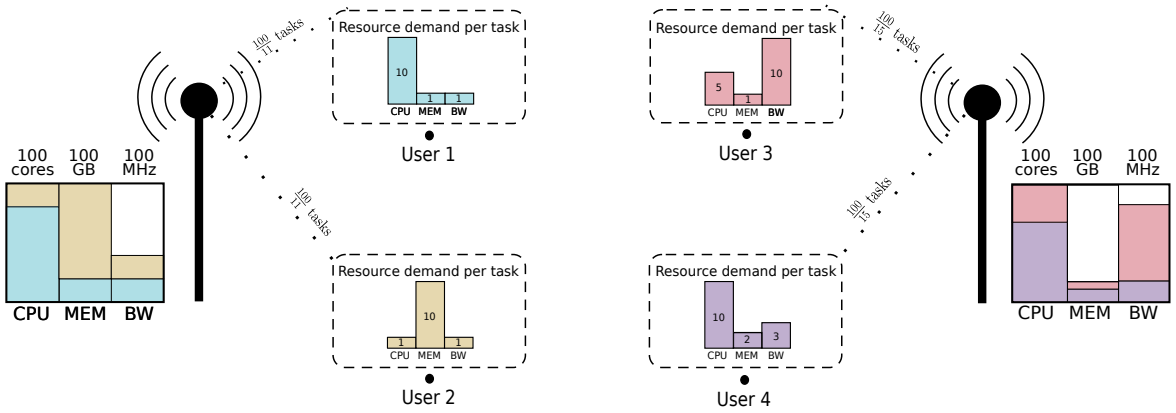


Fig. 1: Example system of 4 users, 2 servers with KS as their local allocation rule, and 3 resources.

TABLE I: Notations.

Symbol	Explanation
n	number of users
m	number of servers
l	number of resources
c_r	capacity of resource r on each server
$\mathbf{c}$	capacity vector, $(c_1, \ldots, c_l)$
$d_{j,r}$	user j's demand for resource r per one task execution
$\mathbf{d}_j$	user j's demand vector, $(d_{j,1}, \ldots, d_{j,l})$
$b_{j,s,r}$	resource r's allocation to user j on server s
$\mathbf{b}_{j,s}$	vector of allocated resources on server s to user j, $(b_{j,s,1}, \ldots, b_{j,s,l})$
$u_j(.)$	user j's utility function, *i.e.*, the number of tasks that user j can execute with its allocation ($u_j(\mathbf{b}_{j,s}) = \min_{r\in[l]}\left\{\frac{b_{j,s,r}}{d_{j,r}}\right\}$)
$\Pi_m([n])$	set of all m-partitions of the set of users $[n]$
A_s	set of users associated with server s
$\mathcal{A}$	Association of users to servers, *i.e.*, $(A_1, \ldots, A_m) \in \Pi_m([n])$
$\lambda_j(A_s)$	utility (*i.e.*, number of tasks) that allocation rule λ allocates to user $j \in A_s$
ρ_j	user j's associated server
$e_{i,j}(\mathcal{A})$	envy of user i for user j's allocation with association $\mathcal{A}$
ψ_j	maximum utility/task that user j can have if a server is entirely allocated to it, *i.e.*, $\min_{r\in[l]}\{c_r/d_{j,r}\}$

Definition 2 (Local Sharing Incentive (LSI))**.** *A mechanism with local allocation rule λ satisfies LSI, if for any $n, m, l \in \mathbb{N}$, any $\mathbf{c} \in \mathbb{R}^l_{++}$, any non-empty $A_s \subseteq [n]$, and any demand profile $\{\mathbf{d}_j\}_{j\in A_s} \in \mathbb{R}^{n\times l}_{++}$, the utility that each user $i \in A_s$ receives is at least as much as the utility it receives when it equally shares its server with all the other users in A_s, i.e.,*

$$\lambda_i(A_s) \geq u_i\left(\mathbf{p}_{i,s}\right) = \min_{r\in[l]}\left\{\frac{c_r/|A_s|}{d_{i,r}}\right\}, \tag{3}$$

where $\mathbf{p}_{i,s} = (c_1/|A_s|, \ldots, c_l/|A_s|)$ denotes the allocation vector that user j receives on server s in the equal share.

Definition 3 (Local Pareto Optimality (LPO))**.** *A mechanism with local allocation rule λ satisfies LPO, if for any $n, m, l \in \mathbb{N}$, any $\mathbf{c} \in \mathbb{R}^l_{++}$, any non-empty $A_s \subseteq [n]$, and any demand profile $\{\mathbf{d}_j\}_{j\in A_s} \in \mathbb{R}^{n\times l}_{++}$, it is impossible to reallocate resources of server s and strictly increase the utility of some user $i \in A_s$ without strictly decreasing the utility of the other users in A_s.*

B. Fairness Across Servers

With LEF and LSI, a mechanism provides absolute envy-freeness and performance isolation between any two users that are associated with the same server. When $m > 1$, we require similar properties for users that are associated with different servers under an NE associations.

Inspired by the study in [21], we consider the notion of envy among any two users at the NE associations of the game induced by the allocation rule λ. Given an NE association $\mathcal{A}^* = (A^*_1, \ldots, A^*_m)$, the envy of user i for user j is

$$e_{i,j}(\mathcal{A}^*) = \frac{u_i\left(\mathbf{b}_{j,\rho^*_j}\right)}{\lambda_i(A^*_{\rho^*_i})} = \frac{\min_{r\in[l]}\left\{\frac{\lambda_j(A^*_{\rho^*_j})d_{j,r}}{d_{i,r}}\right\}}{\lambda_i(A^*_{\rho^*_i})}, \tag{4}$$

where $\mathbf{b}_{j,\rho^*_j} = \left(\lambda_j(A_{\rho^*_j})d_{j,1}, \ldots, \lambda_j(A_{\rho^*_j})d_{j,l}\right)$ is the resource vector that is allocated to user j on server ρ^*_j. User i does not envy user j under association $\mathcal{A}^*$ if and only if $e_{i,j}(\mathcal{A}^*) \leq 1$.

In an absolute envy-free allocation no user envies another user's allocation. However, Example 1 shows that total envy-freeness is incompatible with LPO at the NEs of any mechanism.

Example 1 (Incompatibility of absolute envy-freeness with LPO and NE)**.** *Consider an environment with 3 identical users and 2 identical servers. At NEs of any mechanism that satisfies LPO, one of the users receives a server entirely to itself and the other two users share the other server. In this allocation, there always exists a user on the second server that does not receive more than half of a server and envies the allocation of the first user. Therefore, it is impossible to always satisfy absolute envy-freeness at the NEs of a mechanism that satisfies LPO.*

Since absolute envy-freeness at NEs is not compatible with LPO, we propose α-approximate envy-freeness which ensures that $e_{i,j}(\mathcal{A}^*) \leq \alpha$ for any i and j.

Definition 4 (α-Approximate Envy-Free at NEs (αEF-NE)). *A mechanism with local allocation rule λ that always admits an NE association satisfies αEF-NE, if for any $n, m, l > 0$, any demand profile $\{\mathbf{d}_j\}_{j\in[n]} \in \mathbb{R}_{++}^{n\times l}$, and any NE association $\mathcal{A}^* = (A_1^*, \ldots, A_m^*) \in \Pi_m([n])$, the envy between any two arbitrary users $i, j \in [n]$ is no more than α, i.e., $e_{i,j}(\mathcal{A}^*) \leq \alpha$.*

Inspired by the notion of maximin share in fair allocation of goods [17]–[20], we propose MaxiMin Share Guarantee at NEs (MMS-NE), which is a straightforward generalization of the popular cut-and-choose protocol in the cake-cutting problem. Suppose we ask a user j to partition the resources into n bundles, one bundle for each user, with the condition that the other $n-1$ users get to choose a resource bundle before it. In the worst case, user j receives its least preferred bundle. Consequently, user j will choose a partition to maximize the utility of its least preferred bundle. This maximum possible value is called user j's maximin share value. In our model, it is easy to show that user j's maximin share value is equal to $\frac{1}{\lceil n/m \rceil} \min\limits_{r\in[l]} \left\{ \frac{c_r}{d_{j,r}} \right\}$. Therefore, we propose the following property.

Definition 5 (MaxiMin Share Guarantee at NEs (MMS-NE)). *A mechanism with local allocation rule λ that always admits an NE association satisfies MMS-NE, if for any $n, m, l \in \mathbb{N}$, any $\mathbf{c} \in \mathbb{R}_{++}^{l}$, any demand profile $\{\mathbf{d}_j\}_{j\in[n]} \in \mathbb{R}_{++}^{n\times l}$, and any NE association $\mathcal{A}^* = (A_1^*, \ldots, A_m^*) \in \Pi_m([n])$, the utility that each user receives is at least as much as its maximin share value, i.e., for any $s \in [m]$ and any $j \in [n]$,*

$$\lambda_j(A_{\rho_j^*}^*) \geq \frac{1}{\lceil n/m \rceil} \min_{r\in[l]} \left\{ \frac{c_r}{d_{j,r}} \right\}. \tag{5}$$

C. Challenges in Designing a Fair Mechanism

It is easy to satisfy the properties mentioned above in a single-resource allocation problem by associating a balanced number of users (*i.e.*, $\lfloor n/m \rfloor$ or $\lceil n/m \rceil$) to each server and equally splitting the resource of each server among its associated users. However, designing a mechanism that always admits an NE while satisfying the local properties is an intricate task in the multi-resource allocation problem. Here we discuss several well-known allocation rules and explain why they fail.

Consider the simplest local allocation rule that equally splits resources among the users, *i.e.*, any user j receives $1/|A_{\rho_j}|$ of its associated server. Hence, any association $\mathcal{A} = (A_1, \ldots, A_m)$ such that $\lfloor n/m \rfloor \leq |A_s| \leq \lceil n/m \rceil$ for all $s \in [m]$ is an NE association. Furthermore, with this allocation rule, the users that are associated with the same server receive the exact same allocation. Hence, it satisfies LEF. Moreover, it is trivial that this mechanism satisfies LSI, MMS-NE, and $\frac{\lceil n/m \rceil}{\lfloor n/m \rfloor}$EF-NE. However, it is obvious that this mechanism is highly inefficient in multi-resource environments and violates LPO.

Competitive Equilibrium from Equal Income (CEEI) [3], [32], Proportionally Fair (PF) allocation [33], and Nash Bargaining Solution (NBS) [27] are three other widely used allocation rules that are known to satisfy envy-freeness, sharing incentive, and Pareto optimality in a single-server environment [7]. Thus, with them as the local allocation rule we can satisfy LEF, LSI, and LPO. In our system model, these three allocation rules are equivalent to each other and maximize the product of the users' utilities. Unfortunately, these allocation rules fail to provide a mechanism that guarantees the existence of an NE association. For instance, consider an environment with 4 users, 2 servers, and 3 resources with capacity 1. Let the users' demand vectors be as follows.

$$\mathbf{d}_1 = (1.0, 0.1, 0.1), \quad \mathbf{d}_2 = (0.1, 1.0, 0.1),$$
$$\mathbf{d}_3 = (0.5, 0.1, 1.0), \quad \mathbf{d}_4 = (1.0, 0.2, 0.3).$$

With CEEI, PF, or NBS as the local allocation rule, whenever user 2 or 3 are on user 4's server, they can improve their utility by changing their server association. Moreover, when users 2 and 3 are on the same server, and users 1 and 4 are on the other server, user 4 can improve its utility by moving to the other server. Furthermore, when users 1, 2, and 3 are on the same server and user 4 is on the other server, user 1 is willing to change its server and increase its utility. Hence, in all associations, there exists at least one user who wants to change its server. Therefore, in this environment, there exists no NE association for mechanisms that use CEEI, PF, or NSB as their local allocation rule.

Even dropping the notion of NE does not yield a straightforward mechanism design for satisfying the fairness properties altogether. For instance, the naive extension of KS (resp. NBS) that first associates a balanced number of users to each server and then uses KS (resp. NBS) as the local allocation rule fails to guarantee a good approximation for envy-freeness. Consider $m = 2$ servers and $l > 1$ resources. Let $\mathbf{\Delta}^k$ denote the demand vector of a user whose demand for resource k is 1 and its demand for all other resources is $\delta > 0$. Consider $l + 2$ users with demand vector $\mathbf{\Delta}^1$, and $l - 1$ users with demand vectors $\mathbf{\Delta}^2, \mathbf{\Delta}^3, \ldots, \mathbf{\Delta}^l$. A valid balanced association that naive KS and NBS may end up with is associating $l + 1$ users with demand $\mathbf{\Delta}^1$ to the first server and the other l users to the other server. It is easy to show that $\max\limits_{i,j\in[n]} e_{i,j} = \frac{l+1}{1+\delta(l-1)}$ for this association. Therfore, the envy between the users can grow to $l + 1$ as δ goes to zero.

V. The Multi-resource Allocation Game Induced by Kalai-Smorodinsky Bargaining Solution

In this section, we present the proposed MAGIKS mechanism. MAGIKS always admits an NE association. Furthermore, we prove that under discrete resource demands, MAGIKS finds an NE association in polynomial time for any fixed server configuration. We prove that the allocation derived by MAGIKS always satisfies LEF, LSI, LPO, 2EF-NE, and MMS-NE. It is worth mentioning that MAGIKS always has an NE association and satisfies these properties even with non-discrete resource demands.

A. MAGIKS's Design

The KS allocation rule, which we denote by μ in this paper, is an egalitarian rule that finds the lexicographic maxmin fair solution after a certain normalization of utilities. Let ψ_j denote the maximum utility that user j can have if a server is entirely allocated to it, *i.e.*,

$$\psi_j = \min_{r\in[l]} \{c_r/d_{j,r}\}. \tag{6}$$

Note that the servers are homogeneous and ψ_j is independent of the server identity. The KS allocation rule would normalize each user's utility with respect to its maximum utility. It maximizes these normalized utilities while keeping them equal. Let A_s be the set of users associated with server s. The solution of problem (7) gives the utilities that users in A_s receive with the KS allocation rule.

$$\max_{g,\, x_j} \quad g \tag{7a}$$

$$\text{s.t.} \quad \frac{x_j}{\psi_j} = g, \quad \forall j \in A_s, \tag{7b}$$

$$\sum_{j\in A_s} x_j d_{j,r} \le c_r, \quad \forall r \in [l]. \tag{7c}$$

The value of the equalized normalized utility is represented by g in problem (7).

Let the vector $\tilde{\mathbf{d}}_j = \left(\tilde{d}_{j,1}, \ldots, \tilde{d}_{j,l}\right)$ be the scaled demand vector of user j in which each resource demand is scaled with respect to the resource capacity (*i.e.*, $\tilde{d}_{j,r} = d_{j,r}/c_r$ for all $r \in [l]$). Thus, $\tilde{\mathbf{d}}_j$ represents what portion of each server's resource is required per unit of utility. In other words, $\tilde{d}_{j,r}$ represents how much user j is bottlenecked by resource r. Let $\bar{\mathbf{d}}_j = \left(\bar{d}_{j,1}, \ldots, \bar{d}_{j,l}\right)$ be the normalized demand vector of user j, where

$$\bar{d}_{j,r} = \frac{\tilde{d}_{j,r}}{\max\limits_{r'\in[l]} \left\{\tilde{d}_{j,r'}\right\}} = \frac{d_{j,r}/c_r}{\max\limits_{r'\in[l]} \{d_{j,r'}/c_{r'}\}}. \tag{8}$$

Therefore, the elements in $\bar{\mathbf{d}}_j$ follow the same proportion as the elements in $\tilde{\mathbf{d}}_j$ and are scaled so that the maximum element in $\bar{\mathbf{d}}_j$ is 1.

With simple algebraic manipulation of problem (7), we can show that the utility that user j receives with the KS allocation rule, *i.e.*, $\mu_j(A_s)$, is

$$\mu_j(A_s) = \frac{\psi_j}{l(A_s)}, \ \forall j \in A_s, \tag{9}$$

where $l(A_s)$ is given by

$$l(A_s) = \max_{r\in[l]} \left\{ \sum_{i\in A_s} \bar{d}_{i,r} \right\}, \tag{10}$$

and we refer to it as the load induced by A_s.

In MAGIKS, we use the KS allocation rule as the local allocation rule of each server. At first, each user randomly chooses a server. The servers use equation (9) to feedback to the users the utility they receive based on their server choice. After each move, each user inquires of the utility it can receive if it changes its association to another server. Let ρ_j^k denote user j's server association after move k and A_s^k denote the set of users associated with server s after move k. Let $\mathcal{A}^k = \left(A_1^k, \ldots, A_m^k\right)$ be the association after move k. In each move k, one randomly chosen unhappy user changes its server association to some server s that gives it better utility, *i.e.*,

$$\mu_j\left(A^k_{\rho_j^k}\right) < \mu_j\left(A_s^k \cup \{j\}\right). \tag{11}$$

Note that each time a user moves, some new unhappy users could be created on the moving user's new server and some old unhappy users on its old server may become happy. The users repeat this process until they reach an NE association.

Before proving the existence of an NE association and presenting the convergence time analysis of MAGIKS, we make the following observations on the load function $l(A_s)$ and its relation to the utility that a user receives by the KS allocation rule.

Observation 1. *A user can increase its utility by changing its server association from s to s' if and only if the current load on server s is strictly greater than the load on server s' after moving the user to server s'.*

$$\mu_j\left(A_{s'} \cup \{j\}\right) > \mu_j\left(A_s\right) \iff l\left(A_{s'} \cup \{j\}\right) < l\left(A_s\right).$$

Observation 2. *Since the demand vectors are assumed to be positive (i.e., $d_{j,r} > 0$ for all $j \in [n]$ and $r \in [l]$), the server load is a monotonically increasing function, i.e., for any $S, T \subseteq [n]$ such that $S \subset T$,*

$$l\left(S\right) < l\left(T\right).$$

Observation 3. *Observations 1 and 2 imply that if user j could increase its utility by moving from server s to s', then the original load on server s must have been strictly larger than that of server s', i.e.,*

$$\mu_j\left(A_{s'} \cup \{j\}\right) > \mu_j\left(A_s\right) \Rightarrow l\left(A_{s'}\right) < l\left(A_s\right).$$

B. NE Existence and Convergence Time Analysis

It is easy to show that MAGIKS is not a potential game. Therefore, it is non-trivial to study its NE and convergence properties.

MAGIKS is closely related to the vector scheduling game in [29]. In a vector scheduling game, each selfish job wants to submit its multi-dimensional load vector to a server that gives it the minimum cost. The cost of a server is the maximum element in the sum of load vectors on it. By Observation 1, MAGIKS can be interpreted as a vector scheduling game in which a job's load vector is equal to a user's normalized demand vector and a job's cost function is equal to the load of a user's associated server.

Let us denote the sorted load vector after the k_{th} move by $\mathbf{l}^k_{\text{sorted}} = \left(l(A^k_{\delta_1^k}), \ldots, l(A^k_{\delta_m^k})\right)$, where $\left(\delta_i^k, \ldots, \delta_m^k\right)$ is a permutation of $[m]$ such that $l(A^k_{\delta_1^k}) \ge \ldots \ge l(A^k_{\delta_m^k})$. In the following lemma, we show that the sorted load vector is lexicographically reduced after each move in MAGIKS. A similar claim for the vector scheduling game was proved in [29]. The interested reader may refer to that work for the proof.

Lemma 1. *(Corollary of Theorem 1 in [29]) There is a lexicographical reduction in the sorted load vector after each move in MAGIKS, i.e., there exists some $p \in [m]$ such that $l(A^{k+1}_{\delta^{k+1}_q}) \leq l(A^k_{\delta^k_q})$ for all $q \leq p$ and $l(A^{k+1}_{\delta^{k+1}_p}) < l(A^k_{\delta^k_p})$.*

Theorem 1. *For any $n, m, l \in \mathbb{N}$, any $\mathbf{c} \in \mathbb{R}^l_{++}$, and any demand profile $\{\mathbf{d}_j\}_{j\in A_s} \in \mathbb{R}^{n\times l}_{++}$, MAGIKS always admits an NE association and it can be reached after a finite number of moves.*

Proof. By Lemma 1, $\mathbf{l}^k_{\text{sorted}}$ is lexicographically greater than $\mathbf{l}^{k+1}_{\text{sorted}}$. Since the number of possible sorted load vectors is finite, this game always converges after a finite number of moves. At convergence, the users no longer wish to change their server association, which implies that the game converges to an NE association. □

The question of convergence time to an NE remained open for the vector scheduling game [29], [30]. In this work, we present a time complexity analysis of finding an NE in MAGIKS. Under assumptions that each resource r has some $h_r \in \mathbb{N}$ equal levels, and the users are restricted to choosing one of these levels as their demands for resource r, we prove that for any fixed server configuration, MAGIKS converges to an NE association after $\mathcal{O}(\text{poly}(n))$ moves. Note that this assumption holds for real-world cluster managers such as SLURM [34] and BORG [35].

First, we prove the claim for the case $m = 2$ in Lemma 2. Next, in Theorem 2, we prove the claim for any $m \in \mathbb{N}$ by induction on m.

Lemma 2. *Fix an environment with $m = 2$ servers and some $l \in \mathbb{N}$ resources and some $\mathbf{c} \in \mathbb{R}^l_{++}$ such that each resource r has some $h_r \in \mathbb{N}$ equal levels. Let $n \in \mathbb{N}$ be any arbitrary number of users and $(\mathbf{d}_j)_{j\in[n]} \in \mathbb{R}^{n\times l}_{++}$ be any arbitrary feasible demand profile (i.e., $0 < d_{j,r} \leq c_r$) such that each users can choose one of the h_r levels of resource r as their demand for resource r. MAGIKS converges to an NE association after $\mathcal{O}(n)$ moves.*

Proof. Without loss of generality, we can scale the servers' resource capacity, and the users' demands accordingly, to make $c_r = h_r$ and $d_{j,r} \in \{1, \ldots, h_r\}$. Suppose in move $k+1$, a user j improves its utility by changing its server from s to s'. Since the demands are non-zero, each move changes the load on both of the servers. Moreover, by Lemma 1, we have that $\mathbf{l}^k_{\text{sorted}}$ is lexicographically greater than $\mathbf{l}^{k+1}_{\text{sorted}}$. Therefore, each move reduces the value of the maximum load (*i.e.*, $l(A^{k+1}_{\delta^{k+1}_1}) < l(A^k_{\delta^k_1})$).

Now, we find a lower bound for the reduction in the value of the maximum load that is independent of the value of n and demands. By (10) and (8), we have

$$\begin{aligned} &l(A^k_{\delta^k_1}) - l(A^{k+1}_{\delta^{k+1}_1}) \\ &= \max_{r\in[l]} \left\{ \sum_{i\in A^k_{\delta^k_1}} \bar{d}_{i,r} \right\} - \max_{r\in[l]} \left\{ \sum_{j\in A^{k+1}_{\delta^{k+1}_1}} \bar{d}_{j,r} \right\} \qquad (12) \\ &= \sum_{i\in A^k_{\delta^k_1}} \frac{d_{i,r_1}/h_{r_1}}{\max_{r'\in[l]}\{d_{i,r'}/h_{r'}\}} - \sum_{j\in A^{k+1}_{\delta^{k+1}_1}} \frac{d_{j,r_2}/h_{r_2}}{\max_{r'\in[l]}\{d_{j,r'}/h_{r'}\}}, \qquad (13) \end{aligned}$$

where r_1 and r_2 are the maximizers of the two terms in (12).

Note that $d_{j,r} \leq h_r$, and $d_{j,r}, h_r \in \mathbb{N}$ for any $j \in [n]$ and $r \in [l]$. Therefore, we can always rewrite (13) as

$$l(A^k_{\delta^k_1}) - l(A^{k+1}_{\delta^{k+1}_1}) = \sum_{i\in A^k_{\delta^k_1}} \frac{a_i}{a'_i} - \sum_{j\in A^{k+1}_{\delta^{k+1}_1}} \frac{f_j}{f'_j}, \qquad (14)$$

for some $a_i, a'_i, f_j, f'_j \in \mathbb{N}$ such that $a_i < a'_i$ and $f_j < f'_j$, and $a'_i, f'_j \leq \max_{r,r'\in[l]} \{(h_r - 1)h_{r'}\} = \gamma_1$ for all $i \in A^k_{\delta^k_1}$ and $j \in A^{k+1}_{\delta^{k+1}_1}$. Note that γ_1 is a constant.

All the terms in (14) are rational numbers. Hence, the final value must be a rational number which we denote by $\frac{u}{v}$ with $u \in \mathbb{Z}$, $v \in \mathbb{N}$, and v is no more than the least common multiple of all a'_i's and f'_j's. Hence, v is no more than the least common multiple of all numbers from 1 to γ_1, which we denote by γ_2. Furthermore, we have that $l(A^k_{\delta^k_1}) - l(A^{k+1}_{\delta^{k+1}_1}) = \frac{u}{v} > 0$. Hence, $u \geq 1$ and $l(A^k_{\delta^k_1}) - l(A^{k+1}_{\delta^{k+1}_1}) \geq \frac{1}{\gamma_2}$.

Finally, since the value of the maximum load is at most n and at least 0, the number of improvement moves is no more than $\gamma_2 n$. Note that γ_2 is a constant and independent of the value of n and demands. Therefore, MAGIKS converges to an NE association after $\mathcal{O}(n)$ number of moves. □

Theorem 2. *Fix an environment with some $m, l \in \mathbb{N}$ and some $\mathbf{c} \in \mathbb{R}^l_{++}$ such that each resource r has some $h_r \in \mathbb{N}$ equal levels. Let $n \in \mathbb{N}$ be any arbitrary number of users and $(\mathbf{d}_j)_{j\in[n]} \in \mathbb{R}^{n\times l}_{++}$ be any arbitrary feasible demand profile (i.e., $0 < d_{j,r} \leq c_r$) such that each user can choose one of the h_r levels of resource r as its demand for resource r. MAGIKS converges to an NE association after $\mathcal{O}(n^{m-1})$ number of moves.*

Proof. When there are more than two servers in the environment, even though each improvement move lexicographically reduces the sorted load vector, not all improvement moves necessarily reduce the maximum load value. Therefore, in the proof of this theorem, we have to take into account the number of moves that do not reduce the maximum load value.

We prove this theorem by induction. Analogous to the proof of Lemma 2, without loss of generality, we scale the servers' resource capacity, and the users' demands accordingly, to make $c_r = h_r$ and $d_{j,r} \in \{1, \ldots, h_r\}$. By Lemma 2, the claim holds for the base case $m = 2$. We assume that the claim is true for $m = t - 1$, and we prove it for $m = t$.

The following is an outline of the proof of the induction step. We claim that if the value of the maximum load does not change through moves k to k', there exists a server with the maximum value load that has never lost or received a user through moves k to k'. Further, we use this claim together with the hypothesis of the induction to prove that the number of moves in between two consecutive maximum load reducing moves (MLR-moves) is no more than $\mathcal{O}(n^{t-2})$. Finally, we propose a lower bound for the maximum load value reduction which is independent of t and demands in order to find the maximum number of possible MLR-moves.

Now we are ready to present the proof details.

Claim 1. Let the maximum load remain unchanged from move k to k' (*i.e.*, $l(A^{k'-1}_{\delta_1^{k'-1}}) = \ldots = l(A^k_{\delta_1^k})$). Let S^v denote the set of servers with maximum load value right after the v_{th} move. We claim that none of the servers in $S^{k'-1}$ have participated in any move between k and k', *i.e.*, for all $s \in S^{k'-1}$,

$$A^v_s = A^k_s,\ \forall k \leq v < k'.$$

Proof of claim. Let $S^v = \arg\max_{s\in[m]} \{l(A^v_s)\}$ be the set of servers with maximum load value right after the v_{th} move. It is impossible for the servers in S^k to receive a new user at some move without losing one of its own users prior to that move. Otherwise, there would be lexicographical increase on the sorted load vector, which contradicts the improving nature of the move. Let $s \in S^k$ be any arbitrary server that loses a user for the first time after some move. We claim that server s will never again gain enough load to become a server with maximum load. By Observation 1, whenever server s receives some user from a server with larger load α, its new load will be strictly less than α. Hence, the new load on server s will always remain less than the maximum load value, which is unchanged from move k to k'. Consequently, any server in S^k that lost or received a user at some moves between k and k', is not a member of $S^{k'-1}$. Similarly, we can show that those users who were not a member of S^k cannot gain enough load to become a member of $S^{k'-1}$. Therefore, $S^{k'-1} \subseteq \ldots \subseteq S^k$, and the user association on any server $s \in S^{k'-1}$ remained unchanged through moves k to $k'-1$.

Claim 2. Let k and k' be two consecutive MLR-moves with $k' > k$. There always exists a server that has not been part of any moves between moves k and k', while attaining the maximum load value among all servers.

Proof of claim. Since k and k' are two consecutive MLR-moves, the maximum load value must have remained unchanged between these moves, *i.e.*, $l(A^{k'-1}_{\delta_1^{k'-1}}) = \ldots = l(A^k_{\delta_1^k})$. Let S^v denote the set of servers with maximum load value right after the v_{th} move. We must have $|S^{k'-1}| = 1$, since otherwise the k'_{th} move cannot be an MLR-move. Moreover, By claim 1, the server in $S^{k'-1}$ that always had the maximum load value between moves k and k' did not participate in any move.

Claim 3. The maximum number of moves in between two consecutive MLR-moves is in $\mathcal{O}(n^{t-2})$.

Proof of claim. Let k and k' be two consecutive MLR-moves with $k' > k$. By Claim 2, there exists a server s that has not been part of any moves between k and k', while attaining the maximum load value after move $k'-1$. Hence, the game was being played by users in $[n]\backslash\{A^{k'-1}_s\}$ over the servers in $[t]\backslash\{s\}$. By the induction hypothesis, after at most $\mathcal{O}(n^{t-2})$ moves, we would reach an NE association for this subgame, *i.e.*, no user in $[n]\backslash A^{k'-1}_s$ can improve by moving to any server in $[t]\backslash\{s\}$, and we must have the MLR-move in which a user in $A^{k'-1}_s$ leaves server s. Hence, there are at most $\mathcal{O}(n^{t-2})$ moves between two consecutive MLR-moves.

The proofs of the following two claims are similar to that of Claim 3 and are omitted.

Claim 4. If MAGIKS converges to an NE association without the occurrence of any MLR-move, the total number of moves must be $\mathcal{O}(n^{t-2})$.

Claim 5. The number of moves before the first MLR-move and after the last MLR-move is $\mathcal{O}(n^{t-2})$.

Claim 6. There can be at most $\mathcal{O}(n)$ MLR-moves.

Proof of claim. By following the same approach used in the proof of Lemma 2, we can show that the amount of reduction in the value of the maximum load after each MLR-move is no less that $\frac{1}{\zeta}$, where ζ is some constant and independent of the value of n and demands. Since the value of maximum load is no more than n and no less than 0, there are at most $\mathcal{O}(n)$ MLR-moves.

Together, Claims 3, 4, 5, and 6 imply that MAGIKS converges to an NE association after $\mathcal{O}(n^{t-1})$ moves when $m = t$, which completes the induction claim. □

C. Desired Fairness Properties

Since MAGIKS uses the KS local allocation rule, it is easy to show that it satisfies the three local properties.

Theorem 3. *MAGIKS satisfies LEF, LSI, and LPO.*

Proof. In a single-server environment, Ghodsi *et al.* proved that the KS allocation rule is envy-free, provides sharing incentive, and is Pareto optimal [7]. This directly implies that MSGIKS satisfies LEF, LSI, and LPO. □

The properties αEF-NE and MMS-NE evaluate envy-freeness and performance isolation among users that are associated with different servers at the NE. In the following theorems, we show that MAGIKS satisfies 2EF-NE and MMS-NE. Moreover, we show that 2EF-NE is the best that any mechanism that satisfies LPO can achieve at its NE.

Theorem 4. *i) MAGIKS satisfies 2EF-NE. ii) 2EF-NE is the best that any NE guaranteeing mechanism that satisfies LPO can achieve.*

Proof. i) Let $\mathcal{A}^* = (A^*_1, \ldots, A^*_m) \in \Pi_m([n])$ be any arbitrary NE association. If $n \leq m$, each user would get one server entirely to itself. Hence, $i, j \in [n]$, $e_{i,j}(\mathcal{A}^*) = 1 \leq 2$. If $n > m$, all servers are occupied by at least one user. Therefore, the load on each server is no less than 1. For any two arbitrary users i and j that are associated with the same server, since the KS local allocation rule, μ, satisfies LSI, we would have $e_{i,j}(\mathcal{A}^*) \leq 1$. Let $s, s' \in [n]$ be two arbitrary different servers and $i, j \in [n]$ be two arbitrary users such that $i \in A^*_s$ and

$j \in A^*_{s'}$. Let us start with the relationship between envy and servers' load. By definition of $e_{i,j}(\mathcal{A}^*)$ in (4),

$$e_{i,j}(\mathcal{A}^*) = \frac{\min\limits_{r\in[l]}\left\{\frac{\mu_j(A^*_{s'})d_{j,r}}{d_{i,r}}\right\}}{\mu_i(A^*_s)} = \frac{\frac{\psi_j}{l(A^*_{s'})}\min\limits_{r\in[l]}\left\{\frac{d_{j,r}}{d_{i,r}}\right\}}{\frac{\psi_i}{l(A^*_s)}}$$
$$\overset{(a)}{=} \frac{l(A^*_s)}{l(A^*_{s'})}\frac{\min\limits_{r\in[l]}\{c_r/d_{j,r}\}}{\min\limits_{r\in[l]}\{c_r/d_{i,r}\}}\min_{r\in[l]}\left\{\frac{d_{j,r}/c_r}{d_{i,r}/c_r}\right\},$$

where (a) is due to the definition of ψ_j in (6). Let r^* be the minimizer of $\min\limits_{r\in[l]}\{c_r/d_{i,r}\}$. Since $\min\limits_{r\in[l]}\{c_r/d_{j,r}\} \le c_{r^*}/d_{j,r^*}$ and $\min\limits_{r\in[l]}\left\{\frac{d_{j,r}/c_r}{d_{i,r}/c_r}\right\} \le \frac{d_{j,r^*}/c_{r^*}}{d_{i,r^*}/c_{r^*}}$, we have

$$e_{i,j}(\mathcal{A}^*) \le \frac{l(A^*_s)}{l(A^*_{s'})}\frac{c_{r^*}/d_{j,r^*}}{c_{r^*}/d_{i,r^*}}\frac{d_{j,r^*}/c_{r^*}}{d_{i,r^*}/c_{r^*}} = \frac{l(A^*_s)}{l(A^*_{s'})}. \tag{15}$$

Now we show that $l(A^*_s) \le l(A^*_{s'}) + 1$. Since $\mathcal{A}^*$ is an NE association, we have $\mu_i(A^*_s) \ge \mu_i(A^*_{s'} \cup \{i\})$ and Observation 1 implies that $l(A^*_s) \le l(A^*_{s'} \cup \{i\})$. Therefore,

$$l(A^*_s) \le l(A^*_{s'} \cup \{i\}) = \max_{r\in[l]}\left\{\sum_{k\in A^*_{s'}\cup\{i\}} \bar{d}_{k,r}\right\}$$
$$\le \max_{r\in[l]}\left\{\sum_{k\in A^*_{s'}} \bar{d}_{k,r}\right\} + \max_{r\in[l]}\left\{\bar{d}_{i,r}\right\}$$
$$\overset{(b)}{\le} l(A^*_{s'}) + 1, \tag{16}$$

where (b) is due to $\bar{d}_{i,r} = \frac{\tilde{d}_{i,r}}{\max\limits_{r'\in[l]}\{\tilde{d}_{i,r'}\}} \le 1$. Equations (15) and (16) imply that $e_{i,j}(\mathcal{A}^*) \le \frac{l(A^*_{s'})+1}{l(A^*_{s'})} \le 2$. Hence, MAGIKS satisfies 2EF-NE.

ii) We prove this part by using a counter example. Consider an environment with m servers and $m+1$ identical users. Any NE guaranteeing mechanism that satisfies LPO would associate two users with one server and one user with each of the remaining $m-1$ servers. Due to LPO, each user on the last $m-1$ servers receives the entire server. Moreover, there always exists a user on the first server with an allocation no mare than half of a server. Thus, it is impossible to achieve better than 2-approximate envy-freeness together with NE guarantee and LPO. □

Theorem 5. *MAGIKS satisfies MMS-NE.*

Proof. Let $\mathcal{A}^* = (A^*_1, \ldots, A^*_m) \in \Pi_m([n])$ be any arbitrary NE association. We show that the MMS-NE inequality (5) is satisfied for all users in any of the following cases.

Case 1 ($\lceil n/m \rceil = 1$): It is easy to verify that the MMS-NE satisfying inequality (5) is satisfied with equality, since each server receives at most one user and allocates the entire server to it, *i.e.*, , the utility of each user $j \in [n]$ is ψ_j.

Case 2 ($\lceil n/m \rceil = n/m = k > 1$): When $n \ge m$, all servers are occupied by at least one user, otherwise there would exist a user who can increase its utility by moving to the empty server which contradicts the fact that $\mathcal{A}^*$ is an NE association. Hence, $|A^*_s| \ge 1$ for all $s \in [m]$.

Consider any user in any server s with $|A^*_s| \le k$. Since the KS allocation rule, μ, satisfies LSI, the user must satisfy the inequality (3) which implies the MMS-NE inequality (5) when $|A^*_s| \le k$.

We next show that the MMS-NE inequality (5) is satisfied by any user j in any arbitrary server t with $|A^*_t| \ge k+1$. Since $|A^*_s| \ge 1$ for all $s \in [m]$, if $|A^*_t| \ge k+1$, there should exists some server t' such that $|A^*_{t'}| \le k-1$. Since $\mathcal{A}^*$ is an NE association, user j cannot increase its utility by moving to server t'. Thus, $\mu_j(A^*_t) \ge \mu_j(A^*_{t'} \cup \{j\})$. By LSI inequality (3), we have $\mu_j(A^*_{t'} \cup \{j\}) \ge \min\limits_{r\in[l]}\left\{\frac{c_r/|A^*_{t'}\cup\{j\}|}{d_{i,r}}\right\} \ge \min\limits_{r\in[l]}\left\{\frac{c_r/k}{d_{i,r}}\right\}$.

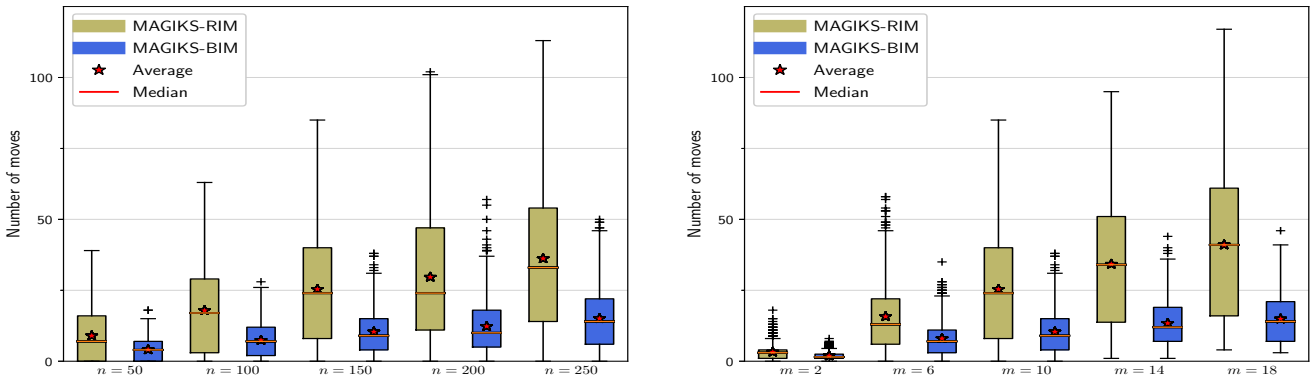
Case 3 ($\lceil n/m \rceil > n/m > 1$): Analogous to Case 2, any arbitrary user in any server s with $|A^*_s| \le \lceil n/m \rceil$ satisfies the MMS-NE inequality (5). Similar to Case 2, we can show that all users in any arbitrary server t with $|A^*_t| > \lceil n/m \rceil$ satisfy the MMS-NE inequality (5) due to the existence of a server t' with $|A^*_{t'}| \le \lceil n/m \rceil - 1$, the fact that $\mathcal{A}^*$ is an NE association, and the LSI property of the KS allocation rule. □

D. Implementation Remarks

It is worth mentioning that MAGIKS can be implemented in a distributed fashion, and users are not required to reveal their sensitive information (*e.g.*, demand vector and server association) to each other. Instead, each user reports its demand vector to the servers to inquire about the number of tasks it could receive if it moves to that server. We assume that the users are truthful and do not game the system by reporting fake demand vectors. The servers respond to this query based on their currently associated users' demand vector. Each unhappy user waits for a random time before sending out a small pilot signal indicating that it wants to move. The first user that sends out this pilot signal will be the one who is allowed to change its server association. Hence, this implementation of MAGIKS requires the users to see each other's pilot signals. However, the pilot signals do not contain any information regarding the identity of the users, their demand vectors, or their server assocaition. We refer to this implementation as MAGIKS with Random Improvement Move (MAGIKS-RIM). Alternatively, servers can implement MAGIKS in a centralized fashion such that in each iteration, they pick the unhappy user who would benefit most from changing its association. We refer to this implementation as MAGIKS with Best Improvement Move (MAGIKS-BIM). By Theorem 2, MAGIKS converges to an NE association in polynomial number of moves for any fixed server configuration, regardless of its improvement move strategy. Hence, both MAGIKS-RIM and MAGIKS-BIM converge to an NE in a polynomial number of moves for any fixed server configuration.

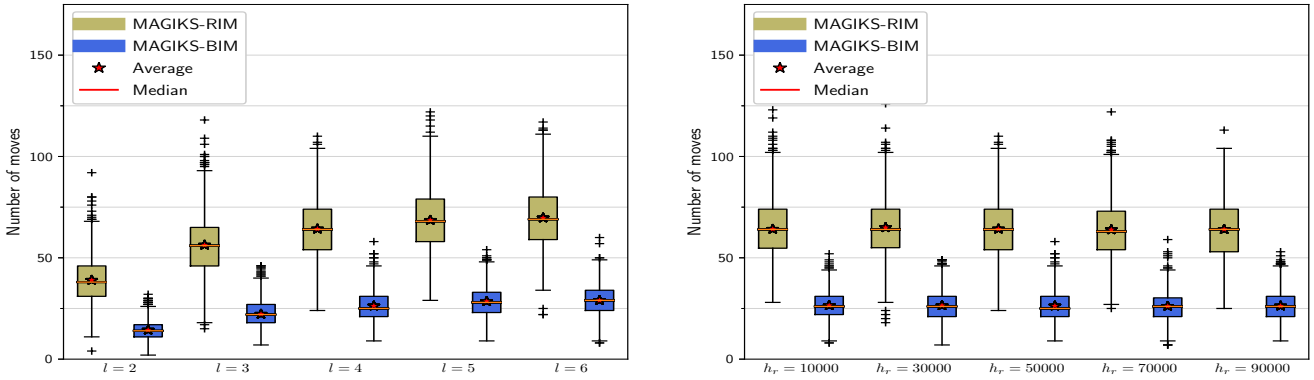
VI. Simulation Results

In addition to proving the NE and fairness properties of MAGIKS in Section V, we further evaluate its performance with both a synthetic dataset and a real-world dataset. Our simulation engine is constructed in Python with packages CVXPY [36], [37] and MOSEK. We choose four benchmark algorithms for resource allocation in our environment:



(a) m is set to 10. (b) n is set to 150.

Fig. 2: Number of moves by MAGIKS with Alibaba cluster trace dataset.



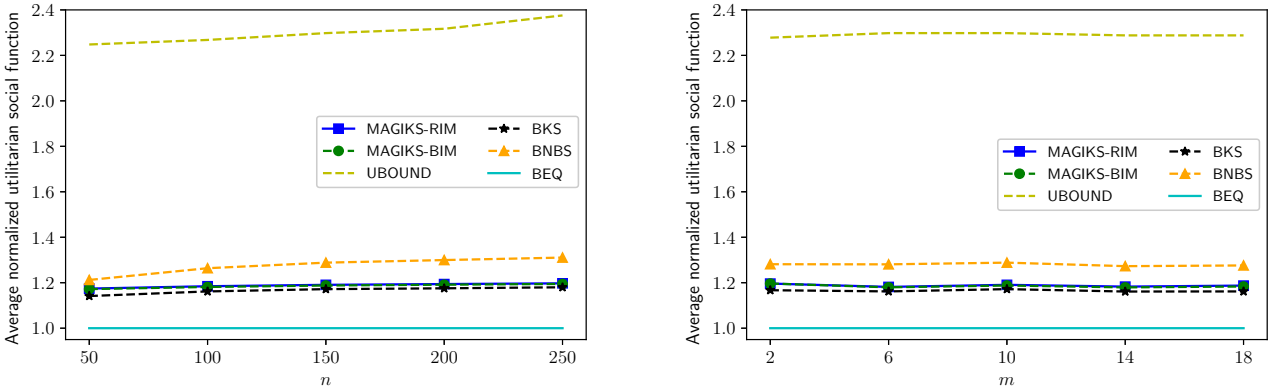
(a) h_r is set to 50000. (b) l is set to 4.

Fig. 3: Number of moves by MAGIKS with synthesized dataset. The number of users, n, and the number of servers, m, are set to 150 and 10, respectively.

- **UBOUND**: It maximizes the Normalized Utilitarian Social Function (NUSF), *i.e.*, $\sum_j \frac{\mu_j(A_{P_j})}{m\psi_j}$, after relaxing this problem by aggregating all servers' resources into a single server.
- **BKS**: We naively extend the DRF/KS-based allocation rules by proposing Balanced KS (BKS) that randomly associates a balanced number of users ($\lfloor n/m \rfloor$ or $\lceil n/m \rceil$) to the servers and uses KS as the local allocation rule.
- **BNBS**: Like BKS, Balanced Nash Bargaining Solution (BNBS) naively extends the NBS-based allocation rules by randomly associating a balanced number of users and using NBS as the local allocation rule.
- **BEQ**: Balanced Equal Division (BEQ) randomly balances the server association and uses equal division of resources as its local allocation rule.

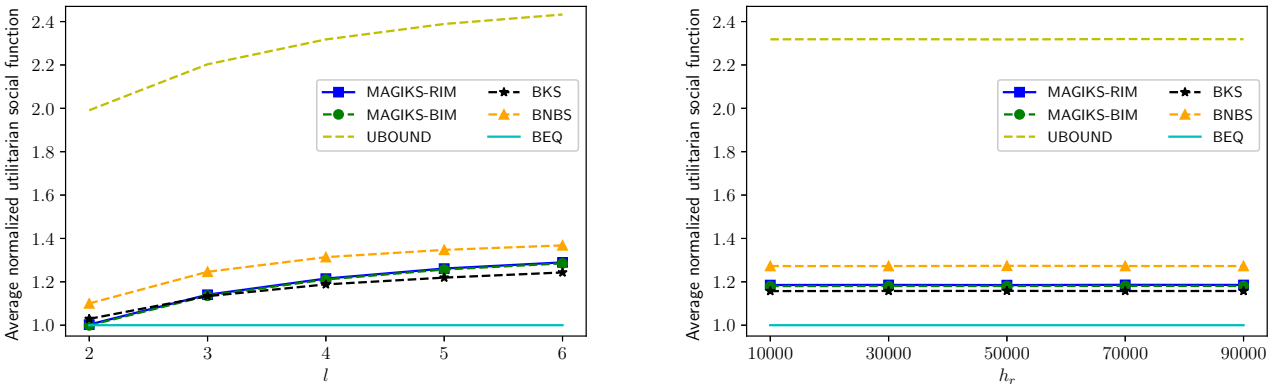
Finding an association that maximizes NUSF is a non-convex and time-consuming problem. Therefore, we study UBOUND, which relaxes this problem and provides an upper bound for the optimal NUSF. BKS, BNBS, and BEQ represent naive extensions of the existing allocation rules. Although BKS and BNBS's association are not necessarily NE associations, these allocation rules are notable for satisfying a subset of the fairness properties. In particular, BKS and BNBS satisfy LEF, LSI, and LPO, and they guarantee the maximin share to users. However, as previously mentioned in Section IV-C, they fail to guarantee a good approximation for envy-freeness, and the envy between two users can be as large as $l+1$ with BKS and BNBS. On the other hand, BEQ's association is an NE association and it satisfies LEF and LSI and guarantees the maximin share to users together with $(1 + 1/\lfloor m/n \rfloor)$ approximate envy-freeness. However, this mechanism is highly inefficient in multi-resource environments and violates LPO. In particular, it provides the minimum NUSF (*i.e.*, 1) among the allocation rules above.

Given any n and m, we generate 100K random realizations where $\mathbf{d}_j$'s are uniformly sampled from the jobs' demand vector in the Alibaba cluster trace dataset. Each job in the Alibaba cluster trace dataset requires resources (*i.e.*, CPU, memory, GPU) in a customized proportion to execute a task. The jobs' demand vectors are included in the trace. We consider each job in the trace as a distinct user in our simulation setting. Like [38]–[41], we synthesize 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 consider the CPU frequency to be 2.6GHz and follow the method of [38]



(a) m is set to 10. (b) n is set to 150.

Fig. 4: Average NUSF versus n and m with Alibaba cluster trace dataset.



(a) h_r is set to 50000. (b) l is set to 4.

Fig. 5: Average NUSF versus l and h with synthesized dataset. The number of users, n, and the number of servers, m, are set to 150, 10, respectively.

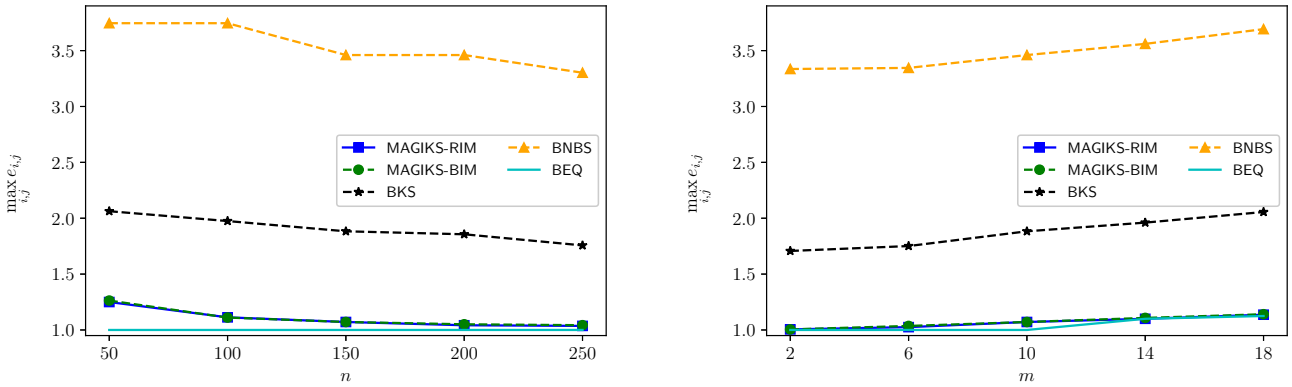
and set the shape and rate parameter of X to 4 and 200, respectively. We consider a general mobile edge computing system with frequency division multiple access with spectral efficiency of $C = 3.5$ and find user j's demand for the wireless communication bandwidth as $d_{j,\text{BW}} = d_{j,\text{bps}}/C$. Furthermore, in each realization, we uniformly sample one server from the set of servers in the Alibaba cluster trace dataset and set it as the capacity vector $\mathbf{c}$ of the m servers in our simulation. The wireless communication bandwidth capacity of the access points is randomly selected from 50, 100, 150, 200, and 250MHz in each realization.

To study the impact of l and h_r, we set n and m to 150 and 10, respectively, and synthesize 100K random realizations where $d_{j,r}$'s are uniformly sampled integers in the interval $[1, h_r]$, and l is uniformly sampled from $\{2, 3, 4, 5, 6\}$. For simplicity, we use the same h_r for all resources. To study the impact of l and h_r on the performance of MAGIKS, we set the other parameter to its default value and changes the value of the parameter under consideration. The default value of l and h_r are 4 and 50000, respectively.

A. Number of Moves to Converge to an NE

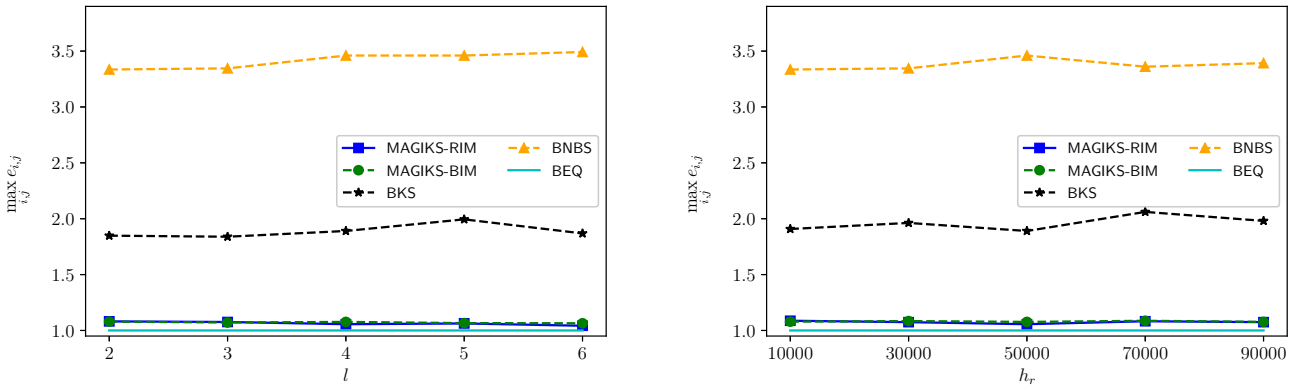
In Figure 2, we illustrate the number of moves taken by MAGIKS-RIM and MAGIKS-BIM to reach an NE versus the number of users and servers in the 100K generated realizations based on the Alibaba cluster trace dataset. Each box represents the interquartile range (IQR), *i.e.*, the distance between the upper and lower quartiles. The whisker represents the largest (resp. smallest) observed point in the range of 1.5×IQR from above (resp. below) the upper (resp. lower) quartile. The whiskers are good approximations for 3 standard deviations of the mean for the normal distribution. Any data point lower (resp. greater) than the lower (resp. upper) bound is represented with a cross. The red bars and red stars represent the median and average value of the data, respectively. Figures 2a and 2b suggest that the average number of moves grows roughly linearly with the number of users and servers. Note that MAGIKS-BIM may require more moves to find an NE in some cases than MAGIKS-RIM. However, our simulation result suggests that, on average, MAGIKS-BIM finds an NE association twice as fast as MAGIKS-RIM.

In Figure 3, we illustrate the number of moves taken by



(a) m is set to 10. (b) n is set to 150.

Fig. 6: Maximum envy versus n and m with Alibaba cluster trace dataset.



(a) h_r is set to 50000. (b) l is set to 4.

Fig. 7: Maximum envy versus l and h with synthesized dataset. The number of users, n, and the number of servers, m, are set to 150, 10, respectively.

MAGIKS-RIM and MAGIKS-BIM to reach an NE versus the number of resources and h_r in the 100K synthesized realizations with 150 users and 10 servers. Introducing more resources to the system reduces the probability of users being bottlenecked on the same resource. Hence, the users will be compatible with each other more often and less willing to change their server association. Thus, the average number of moves shown in Figure 3a becomes insensitive to l for $l \geq 5$. Furthermore, Figure 3b shows that the value of h_r does not impact the average number of moves taken by MAGIKS-RIM and MAGIKS-BIM.

B. Normalized Utilitarian Social Function

In Figures 4 and 5, we evaluate the performance of MAGIKS in terms of NUSF, *i.e.*, $\sum_j \frac{\mu_j(A_{\rho_j})}{m\psi_j}$. In Figure 4a, the Alibaba trace-based simulation results show that MAGIKS's average NUSF is better than 50% of the loose upper bound of the optimal NUSF (*i.e.*, UBOUND). Furthermore, our simulation results suggest that the distributed implementation of MAGIKS does not damage its performance in terms of NUSF compared to its centralized implementation. Since each user's utility is normalized with respect to the sum of the maximum number of tasks that it can execute on each server (*i.e.*, $m\psi_j$), changing the number of servers does not have any impact on the average NUSF.

Increasing the number of resources reduces the conflicts between users and makes it easier for them to compensate for each other's bottleneck. Thus, the average NUSF of MAGIKS increases with the number of resources as shown in Figure 5a. Finally, Figure 5b shows that the average NUSF is not impacted by the value of h_r.

Simulation results show that the naive extension of NBS outperforms MAGIKS in terms of utilization. However, this allocation rule fails to guarantee a good approximation for envy-freeness (see Section VI-C). Moreover, its balanced association is not necessarily an NE association. Unfortunately, unlike KS, NBS cannot be used as a local allocation rule in our multi-resource allocation game, since there exist cases in which no association is an NE association with this allocation rule (see Section IV-C).

C. Envy Between Users

In Figures 6 and 7, we study the maximum envy across the users by recording the maximum observed envy during our 100K random realization of each setting. BEQ finds the most balanced distribution of resource allocation by randomly choosing a balanced association and equally splitting the servers across their associated users. Hence, it guarantees $\frac{\lceil n/m \rceil}{\lfloor n/m \rfloor}$EF-NE, which is the best that any allocation rule that provides maximin share guarantee can achieve. However, as our simulation results in the previous section show, this envy-freeness is achieved through sacrificing the utilization. Furthermore, Figures 6 and 7 confirm our theoretical analysis in Section IV-C and show that BNBS and BKS perform poorly in terms of providing approximate envy-freeness in practice.

VII. Conclusion and Discussion

In this paper, we propose a fair and efficient mechanism for a multi-server environment to allocate multiple resources to multiple users. Each user is a selfish utility-maximizing agent that chooses a single server for its job execution. To the best of our knowledge, this paper is the first to study algorithmic fairness and efficiency of mechanisms that jointly match the servers to the users and share their resources. We propose MAGIKS, which uses KS as its local allocation rule and induces a multi-resource allocation game among the users. Moreover, MAGIKS satisfies local and cross-server fairness properties including LEF, LSI, LPO, 2EF-NE, and MMS-NE. Notably, 2EF-NE is the best approximate envy-freeness that any mechanism that satisfies LPO can achieve. However, these properties rely on a set of assumptions worth discussing.

First, we assume users play truthfully and report their true demand vectors. A mechanism is said to be strategy-proof if no user has an incentive to misrepresent its true preferences (*i.e.*, its true demand vectors). Unfortunately, in our numerical experiments, we have observed that MAGIKS does not satisfy strategy-proofness. However, it satisfies scale-invariance, a weaker version of strategy-proofness, which guarantees no user benefits from reporting a scaled demand vector (*i.e.*, $\alpha\mathbf{d}_j$) instead of its true demand vector (*i.e.*, $\mathbf{d}_j$). An interesting direction for future work is to study the possibility of satisfying strategy-proofness together with the other properties.

Second, we assume that the servers are homogeneous, and relaxing this assumption is a natural direction for future work. In particular, it is not known whether extending MAGIKS to the heterogeneous-server environments would result in a game that always admits an NE association since Lemma 1 does not hold anymore. Furthermore, satisfying MMS seems to be a non-trivial task in heterogeneous-server environments. It would be interesting to study whether such environments always admit an MMS allocation.

Third, in Theorem 2, we prove that under discrete resource demands, MAGIKS finds an NE in $\mathcal{O}\left(n^{m-1}\right)$ moves, which is polynomial in n for any fixed server configuration (*i.e.*, m and h_r for each resource r are some constants and independent of the value of n). Nonetheless, Theorems 1, 3, 4, and 5 indicate that MAGIKS converges to an NE association after a finite number of moves and satisfies LEF, LSI, LPO, 2EF-NE, and MMS-NE even without the assumptions of discrete resource demands and/or fixed server configuration. To the best of our knowledge, this paper is the first to study the time complexity to reach an NE in the vector scheduling game. An interesting direction for future work is to study the tightness of the bound presented in Theorem 2. Our simulation results suggest that the average number of moves of MAGIKS grows roughly linearly with the number of users and servers, and it performs well in terms of the utilitarian social objective. An interesting direction for future work may be lower-bound analysis of MAGIKS's performance in terms of the utilitarian social objective.

Last, we assume that the resources are divisible and the users accept fractional utilities. Under resource indivisibility or integer utilities, LEF, LSI, and 2-EF are no longer satisfiable. An interesting direction for future work is to study the possibility of satisfying the approximate version of these properties under indivisible resources or integer utilities.

References

[1] H. Steihaus, "The problem of fair division," *Econometrica*, vol. 16, pp. 101–104, 1948.

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

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

[4] J. H. Reijnierse and J. A. Potters, "On finding an envy-free pareto-optimal division," *Mathematical Programming*, vol. 83, no. 1, pp. 291–311, 1998.

[5] S. J. Brams, S. J. Brams, and A. D. Taylor, *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.

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

[7] 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.

[8] 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. 3:1–3:22, Mar. 2015.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] ——, "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.

[15] 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.

[16] 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.

[17] D. Kurokawa, A. D. Procaccia, and J. Wang, "Fair enough: Guaranteeing approximate maximin shares," *J. ACM*, vol. 65, no. 2, Feb. 2018.

[18] G. Amanatidis, E. Markakis, A. Nikzad, and A. Saberi, "Approximation algorithms for computing maximin share allocations," *ACM Trans. Algorithms*, vol. 13, no. 4, Dec. 2017.

[19] M. Ghodsi, M. Hajiaghayi, M. Seddighin, S. Seddighin, and H. Yami, "Fair allocation of indivisible goods: Improvements and generalizations," in *Proc. ACM Conference on Economics and Computation (EC)*, 2018, pp. 539–556.

[20] J. Garg and S. Taki, "An improved approximation algorithm for maximin shares," in *Proc. ACM Conference on Economics and Computation (EC)*, 2020, pp. 379–380.

[21] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi, "On approximately fair allocations of indivisible goods," in *Proc. of the fifth ACM Conference on Electronic Commerce*, 2004, pp. 125–131.

[22] 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.

[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] ——, "Sharing incentives and fair division for multiprocessors," *IEEE Micro*, vol. 35, no. 3, pp. 92–100, 2015.

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

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

[27] J. Nash, "The Bargaining Problem," *Econometrica*, vol. 18, no. 2, pp. 155–162, Apr. 1950.

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

[29] D. Ye and J. Chen, "Non-cooperative games on multidimensional resource allocation," *Future Generation Computer Systems*, vol. 29, no. 6, pp. 1345–1352, 2013.

[30] L. Epstein and E. Kleiman, "Scheduling selfish jobs on multidimensional parallel machines," *Theoretical Computer Science*, vol. 694, pp. 42–59, 2017.

[31] L. Libman and A. Orda, "Atomic resource sharing in noncooperative networks," *Telecommunication Systems*, vol. 17, no. 4, pp. 385–409, Aug. 2001.

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

[33] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[34] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60.

[35] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with BORG," in *Proc. of the Tenth European Conference on Computer Systems (EuroSys)*, 2015.

[36] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

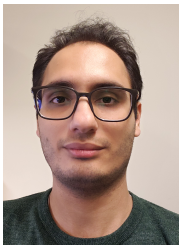
[37] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[38] 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.

[39] 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.

[40] 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.

[41] 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.



Erfan Meskar received the B.Sc. degree in Electrical Engineering from Amirkabir University of Technology, Tehran, Iran, in 2013, and the M.A.Sc. degree in Electrical and Computer Engineering from McMaster University, Hamilton, Ontario, Canada, in 2015. He is currently a Ph.D. student at the Department of Electrical and Computer Engineering at the University of Toronto, Toronto, Ontario, Canada. His current research interests are in algorithmic fairness, algorithmic game theory, and mobile edge computing.

Ben Liang 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.