Capturing Resource Tradeoffs in Fair Multi-Resource Allocation

Doron Zarchy
School of Computer Science and Engineering
Hebrew University of Jerusalem
doronz@cs.huji.ac.il

David Hay
School of Computer Science and Engineering
Hebrew University of Jerusalem
dhay@cs.huji.ac.il

Michael Schapira
School of Computer Science and Engineering
Hebrew University of Jerusalem
schapiram@cs.huji.ac.il

***Abstract*— Cloud computing platforms provide computational resources (CPU, storage, etc.) for running users' applications. Often, the same application can be implemented in various ways, each with different resource requirements. Taking advantage of this flexibility when allocating resources to users can both greatly benefit users and lead to much better global resource utilization. We develop a framework for fair resource allocation that captures such implementation tradeoffs by allowing users to submit multiple "resource demands". We present and analyze two mechanisms for fairly allocating resources in such environments: the Lexicographically-Max-Min-Fair (LMMF) mechanism and the Nash-Bargaining (NB) mechanism. We prove that NB has many desirable properties, including Pareto optimality and envy freeness, in a broad variety of environments whereas the seemingly less appealing LMMF fares better, and is even immune to manipulations, in restricted settings of interest.**

I. Introduction

How to fairly allocate resources to multiple interested parties is an age-old challenge and a prominent research area in game theory, economics, and computer science. Of special interest, from a networking perspective, is the allocation of computational resources (e.g., CPU, memory, storage, bandwidth, etc.) in cloud computing platforms. Indeed, recently there has been a surge on interest in schemes for fairly allocating multiple resources motivated by the allocation of "bundles" of heterogeneous resources in datacenters (see, e.g., [1], [2] and references therein).

Our focus here is on a yet unexplored aspect of resource allocation in large-scale computational environments, e.g., cloud computing platforms. Often, the same computational task can be implemented in several different ways, each with different resource requirements. Consider, e.g., the well-studied tradeoffs in task execution between the amount of CPU and the amount of memory allotted to executing a task [3]. We argue that this flexibility can be of great importance from a fair multi-resource allocation perspective, both from the individual user's perspective and from a global resource utilization perspective.

To see this, consider even the simple toy example in which a *single* user needs to run two identical tasks on the cloud and no other users are competing over the cloud's resources. To execute each of the two tasks, the user needs either a large quantity of CPU and little memory, or a large quantity of memory and little CPU. Specifically, to run a task the user needs either a $(1-\varepsilon)$-fraction of the cloud's CPU and an ε-fraction of the memory *or* an ε-fraction of the CPU a $(1-\varepsilon)$-fraction of the memory. Now, if the user is limited by the cloud tenant-provider interface to only specifying a single resource requirement (as in, e.g., [1]), and chooses to report, say the much-CPU-little-memory requirement, he cannot hope to be able to complete more than a single task (unless the cloud provider's allocation mechanism hurls a huge amount of unrequested memory at the user...). Contrast this with the scenario that the user can specify multiple resource demands corresponding to different task implementations. Now, the user can specify both possible resource-requirements and consequently complete both tasks, as both requirements can be fulfilled concurrently.

This toy example illustrates how exploiting the flexibility afforded by the ability to run different realizations of the same task can lead to a higher utility for the user and better global utilization of resources. These effects can be greatly amplified when there are multiple users with many diverse tasks to run. Exploiting resource-tradeoffs in task implementaion to better user experience and resource utilization is yet another potential gain from rendering datacenters more predictable by extending the tenant-provider interface (see, for instance, [4]).

We formally model cloud computing (and, more generally, multi-resource) environments with resource-tradeoffs. Intuitively, each user is allowed to specify multiple resource-requirements (corresponding to the requirements of different task implementations) and the utility a user derives from the resources allocated to him is the maximum number of tasks he can complete with these resources. We propose and study two different mechanisms, which reflect two different classical economic approaches for fairly allocating resources: the Nash Bargaining (NB) mechanism and the Lexicographically Max-Min Fair (LMMF) mechanism.

We analyze both mechanisms from three main angles:

- **Computational efficiency.** Does the mechanism run in time that is polynomial in the natural parameters, such as the number of users, resources, etc.?
- **Fairness.** Does the mechanism *fairly* allocate resources to users? We consider several well-studied notions of

978-1-4799-8381-0/15/$31.00 ©2015 IEEE

fairness: Pareto optimality, envy-freeness, and max-min fairness.

- **Incentive compatibility.** Are users incentivized to report their true resource requirements to the mechanism, or can a user gain from "lying"?

We analyze NB and LMMF in two opposite environments: (1) when no restrictions whatsoever are imposed on users' resource demands; and (2) when resource-tradeoffs are linear, i.e., when the total amount of resources needed to execute a task is constant, but different combinations of resources are possible (as in the above toy example). We present both positive and negative results for many different desiderata (including computational efficiency, Pareto optimality, envy freeness, sharing incentive, strategyproofness, and more). Our results establish that while NB provides significant benefits in general, LMMF is more appealing when resource-tradeoffs are linear. We view our contributions as the first step in the exploration of how resource-tradeoffs can be leveraged to improve cloud computing platforms. Analyzing other mechanisms and exploring other restrictions on resource-tradeoffs are left as two important directions for future research.

II. Model and Desiderata

A. Model

Users, resources, and resource-demands. A cloud computing environment provides a pool of k computational resources, $R = \{1, \ldots, k\}$. Let C_r denote the available quantity of resource r. A set $N = \{1, ..., n\}$ of users shares the cloud's resource pool. Each user $j \in N$ has a task to perform that can be implemented in M_j different ways. The resource requirements for j's task are thus captured by a set of M_j *resource-demands* $D_j = \{d_{j1}, ..., d_{jM_j}\}$, where each element in D_j is a k-dimensional *demand vector* d_{jm} that specifies the quantity of each of the k resource required for the m'th implementation of the task. E.g., if the set of resources consists of CPU and memory only, an implementation that requires 1 unit of CPU and 3 units of memory is represented by the demand vector (1,3). Let d^r_{jm} denote the r'th resource in demand vector d_{jm}.

Utility functions. Each user j has a *utility function* (or utility, in short) u_j such that, for every vector of resource quantities $X = (X_1, \ldots, X_k)$, $u_j(X)$ specifies the utility user j derives from being allocated these quantities. Our focus here is on the natural "maximum packing" utility function, which captures the number of tasks the user can execute with its allocated resources. Before formally presenting this utility function, consider the example in Figure 1). User 1 has demands $D_1 = \{(1, 2, 1), (0, 1, 3), (2, 0, 2)\}$. Suppose that 1 is allocated the vector of resource quantities $X = (11, 11, 13)$. Then, user 1's utility is 8.5 as this is the maximum amount of tasks user 1 can complete with these resource quantities, computed as follows: $5 \times (1, 2, 1) + 1 \times (0, 1, 3) + 2.5 \times (2, 0, 2) \leq (11, 11, 13)$. To put this formally, if user j is allocated resources $X = (X_1, ..., X_k)$, $u_j(X)$ is the solution to the following linear program:

$$u_j(X) = \max \sum_{m=1}^{M_j} \alpha_m \quad (1)$$
$$\text{subject to} \quad \sum_{m=1}^{M_j} \alpha_m d_{jm} \leq X$$
$$\alpha_m \geq 0 \quad \forall m \in [M_j]$$

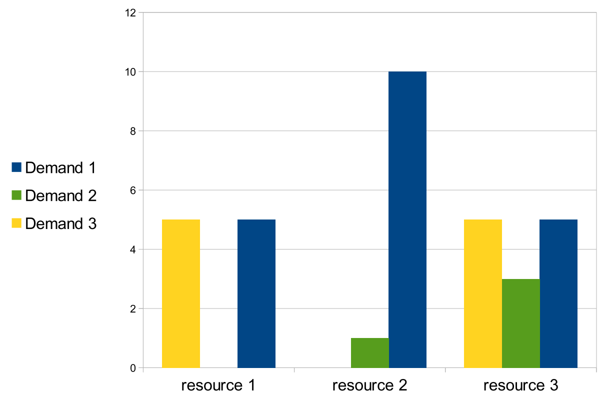

Fig. 1. Consider a resouce pool of $(11, 11, 13)$ and a user with three demand vectors $d_{11} = (1, 2, 1)$, $d_{12} = (0, 1, 3)$, and $d_{13} = (2, 0, 2)$. The optimal "packing" of the user's demands in this resource pool is $5 \times (1, 2, 1) + 1 \times (0, 1, 3) + 2.5 \times (2, 0, 2) \leq (11, 11, 13)$, thus yielding a utility of 8.5, as described above.

Mechanisms. The cloud allocates resources to the users by receiving as input users' resource-demands and then running some resource-allocation *mechanism* to compute the quantity of each resource allocated to each user. All computed allocations must be *feasible*, in the sense that the overall quantities of resources allocated cannot exceed the total amount of resources in the cloud's resource pool. Thus, a mechanism takes as input $D_j = \{d_{j1}, ..., d_{jM_j}\}$ from each user j and allocates, to each user j, a vector of resource quantities $X_j = (X_{j1}, \ldots, X_{jk})$ such that for every resource $r \in [k]$, $\sum_{j \in N} X_{jr} \leq C_r$.

B. Desiderata

We are interested in mechanisms that are (1) computationally-efficient; (2) fair; and (3) incentives. While computational efficiency simply means that the mechanism must run in time that is polynomial in the input parameters—the number of resources k, the number of users n, and the size of each user j's set of resource demands D_j—fairness and incentives require further explanation.

Fairness. We present below three well-studied notions of fairness from economic theory:

- **Pareto-Optimality (PO):** A mechanism is PO if the allocation it outputs is such that in no other allocation does some user have strictly higher utility unless some other user has strictly lower utility, i.e., if the mechanism returns allocation $Y = (Y_1, \ldots, Y_n)$ then in any feasible allocation $X = (X_1, \ldots, X_n)$, if $u_i(X_i) > u_i(Y_i)$ for some user $i \in N$ then $u_j(X_j) < u_j(Y_j)$ for some other user $j \in N$.

- **Envy-Freeness (EF):** A mechanism is EF if it returns allocation $Y = (Y_1, \ldots, Y_n)$ such that no user strictly prefers another user's assigned resources to its own, i.e., for every pair of users $i, j \in N$, $u_i(Y_i) \geq u_i(Y_j)$.
- **Max-Min Fairness (MMF):** A mechanism is MMF if it maximizes the utility of the "least happy" user, i.e., it outputs the allocation $Y = (Y_1, \ldots, Y_n)$ for which the value $\min_{i \in N} u_i(Y_i)$ is maximized.

Incentives We next present two important concepts: strategyproofness and sharing-incentive.

- **Strategyproofness (SP):** A mechanism is SP if no user can benefit by misreporting his resource-demands regardless of other users' reports, i.e., for each user $i \in N$, and for every possible report of resource-demands D_j by every user $j \neq i$, if Y_i is the set of resources allocated to i when i reports his true resource demands D_i, $u_i(Y_i) \geq u_i(X)$ for every set of resources X that i can be allocated by reporting different resource-demands. We point out that some of our positive results actually apply to the stronger notion of "group-strategyproofness" (see, e.g., [5]).
- **Sharing-Incentive (SI):** A mechanism is SI if each user (weakly) prefers the mechanism's allocation to getting a fraction of $\frac{1}{n}$ of each of the resources (his arguably "fair share"), i.e., for every user $j \in N$, $u_j(Y_j) \geq u_j(\frac{1}{n}, ..., \frac{1}{n})$, where Y_j specifies the resource quantities assigned to user j in the mechanism's outputted allocation.

III. Two Mechanisms

We now describe two mechanisms for fairly allocating resources: the Lexicographically Max-Min Fair (LMMF) mechanism and the Nash Bargaining (NB) mechanism. These two mechanisms reflect two different economic approaches to resource allocation and have previously been studied in other contexts (e.g., when each user has a single demand vector). See Section VII for an exposition of related work.

A. The LMMF mechanism

We first present the Max-Min fair (MMF) mechanism, which allocates resources so as to maximize the utility of the "poorest" user (i.e., the user who can complete the lowest number of tasks). To illustrate this, consider the following examples:

Example 1. *A resource pool of* $C = (6, 6, 6)$ *and two users with demands* $D_1 = \{(2, 3, 0), (0, 0, 2)\}$ *and* $D_2 = \{(2, 0, 1), (0, 3, 0)\}$.

Example 2. *A resource pool of* $C = (1, 1, 1)$ *and two user with single demand vector each,* $D_1 = \{(1, 0, 0)\}$ *and* $D_2 = \{(0, \frac{1}{2}, \frac{1}{2})\}$.

In the scenario described in Example 1, MMF returns the allocation $Y_1 = (2, 3, 4)$ and $Y_2 = (4, 3, 2)$. Observe that the

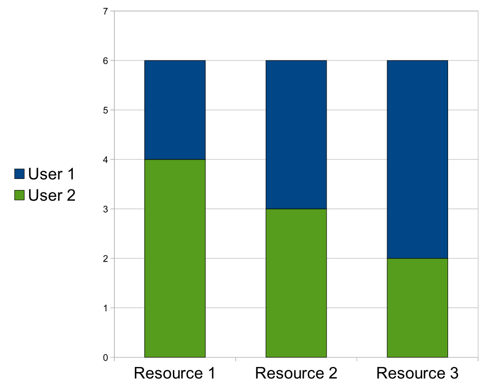

Fig. 2. Max-min fair allocation in Example 1. User 1 gets utility of 3 as it can complete 3 tasks as follows: $1 \times (2, 3, 0) + 2 \times (0, 0, 2)$. User 2 gets utility of 3 as well as it can also complete 3 tasks as follows: $2 \times (2, 0, 1) + 1 \times (0, 3, 0)$.

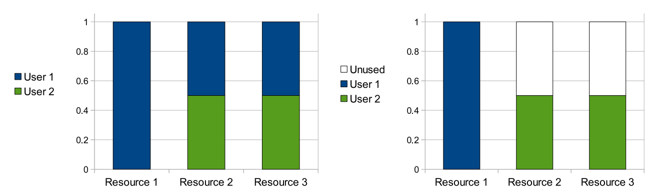

Fig. 3. Example 2. On the right-hand side is a MMF allocation. On the left-hand side is a LMMF allocation.

utility of both users is then 3 (user 1 can "pack" $1 \times (2, 3, 0) + 2 \times (0, 0, 2)$ in his allocated bundle of resources, whereas user 2 can pack $2 \times (2, 0, 1) + 1 \times (0, 3, 0)$). See Figure 2.

However, MMF can sometimes be suboptimal, in the sense that available resources that may benefit users might not be allocated. For instance, in example 2, a possible MMF allocation is $Y_1 = (1, 0, 0), Y_2 = (0, \frac{1}{2}, \frac{1}{2})$. Observe that this allocation is indeed MMF as under any feasible allocation user 1 cannot hope to achieve utility greater than 1. Hence, any allocation under which user 1's utility is 1 and user 2's utility is at least 1 is MMF. However, this allocation is not Pareto optimal, as doubling user 2's resources will increase his utility without harming user 1. This is where *lexicographic* max-min fairness (LMMF) comes in.

To overcome the suboptimality of MMF, LMMF also maximizes the utility of the "poorest" user but, amongst all such allocations, selects the one that maximizes the utility of the second poorest user, and so on. In Example 1, LMMF returns the exact same allocation as MMF. However, in Example 2, LMMF outputs the allocation $Y_1 = (1, 0, 0)$ and $Y_2 = (0, 1, 1)$, which is indeed Pareto optimal. See Figure 3. Th We are now ready to formally define MMF and LMMF.

The Max-Min Fair (MMF) mechanism. Given a resource pool $C = (C_1, \ldots, C_k)$ and users' resource demands, MMF finds an allocation $Y = (Y_1, ..., Y_n)$,

$$\begin{aligned} \text{Maximize} \quad & t \\ \text{Subject to} \quad & u_j(Y_j) \geq t \quad \forall j \in N \\ & \sum_{j \in N} Y_j \leq C \end{aligned} \tag{2}$$

The Lexicographically Max-Min Fair (LMMF) mechanism. To formally present the LMMF mechanism we require the following terminology and notation. For a given allocation $Y = (Y_1, \ldots, Y_n)$, let $\langle Y \rangle$ denote the vector that contains the n elements in $\{u_1(Y_1), \ldots, u_n(Y_n)\}$ sorted in non-decreasing order, i.e., $u_{i_1}(Y_{i_1}) \leq u_{i_2}(Y_{i_2}) \leq \ldots u_{i_n}(Y_{i_n})$. An allocation Y is *lexicographically greater* than another allocation Y', denoted by $\langle Y \rangle \succeq \langle Y' \rangle$, if the first non zero component of $(\langle Y \rangle - \langle Y' \rangle)$ is positive. An allocation vector Y is *lexicographically no less than* Y', denoted by $\langle Y \rangle \succeq \langle Y' \rangle$, if $(\langle Y \rangle - \langle Y' \rangle) = 0$, or the first non-zero component of $(\langle Y \rangle - \langle Y' \rangle)$ is positive. We are now ready to define lexicographic max-min fairness.

Definition 1. An allocation Y is lexicographically max-min fair if $\langle Y \rangle \succeq \langle Y' \rangle$ for every feasible allocation Y'.

The LMMF mechanism outputs, for every input sets of resource-demands, a lexicographically max-min allocation. We now explain how this computation is executed. We prove that the computation indeed terminates in polynomial time and outputs a lexicographically max-min fair allocation in Section IV.

LMMF proceeds in iterations:

Iteration 1: LMMF solves a linear program to compute the maximum value a_1 such that in some feasible allocation $Y = (Y_1, \ldots, Y_n)$ the utility of each and every user is *exactly* a_1.[1] LMMF then checks, for every user, whether his utility in Y cannot be increased without decreasing the utility of other users. All such users are placed in the set p_1.

Iteration 2: LMMF solves a linear program to compute the maximum value a_2 such that in some feasible allocation $Y = (Y_1, \ldots, Y_n)$ the utility of each user in p_1 is exactly a_1 and the utility of all other users is exactly a_2. LMMF then checks, for every user not in p_1, whether his utility in Y cannot be increased without decreasing the utility of users in p_1. All such users are placed in the set p_2.

Iteration t=3,4,. . . : Similarly, LMMF solves a linear program to compute the maximum value a_t such that in some feasible allocation $Y = (Y_1, \ldots, Y_n)$ the utility of each user in p_i for all $i < t$ is exactly a_i and the utility of all other users is exactly a_t. LMMF then checks, for every user not in p_i for $i < t$, whether his utility in Y cannot be increased without decreasing the utility of users in p_i for $i < t$. All such users are placed in the set p_t.

[1] We point out that this can indeed be formulated as a linear program (for the interest of brevity, the formulation is deferred to the full paper.)

This continues until all users are placed in some p_t set, at which point LMMF outputs the allocation Y computed at the last iteration.

To see how LMMF works in a concrete scenario, consider resource pool $C = (1, 1, 1, 1)$ and three users with demands $D_1 = \{(\frac{1}{2}, \frac{1}{2}, 0, 0)\}$, $D_2 = \{(0, \frac{1}{2}, \frac{1}{2}, 0)\}$, and $D_3 = \{(\frac{1}{2}, 0, \frac{1}{2}, 0), (0, 0, 0, 1)\}$. At the first iteration, LMMF solves a linear program to compute $a_1 = 1$, as all users can achieve utility of 1, e.g., in the feasible allocation $Y = ((\frac{1}{2}, \frac{1}{2}, 0, 0), (0, \frac{1}{2}, \frac{1}{2}, 0), (0, 0, 0, 1))$ where users 1 and 2 are each allocated precisely their demand vectors and user 3 is allocated his second demand vector. As users 1 and 2 cannot attain utility higher than 1 in Y (as the second resource is fully utilized) LMMF creates the set $p_1 = \{1, 2\}$. In the next iteration, LMMF solves another linear program to compute $a_2 = 2$. Indeed, consider the allocation resulting from allocating users 1 and 2 the exact same resources as in Y and adding $(\frac{1}{2}, 0, \frac{1}{2}, 0)$ to 3's allocated resources in Y. Observe that again users 1 and 2 have a utility of 1, but now user 3's utility is 2 (=a_2). User 3 is now placed in the set p_2. As all users are now placed in either p_1 or p_2, LMMF terminates and outputs this allocation.

B. Nash Bargaining Mechanism

Consider a scenario in which each user initially has an "endowment" of a $\frac{1}{n}$'th fraction of each resource. Suppose that there are only two resources, each of quantity 1, and two users, 1 and 2, with demands $D_1 = \{(1, 0)\}$ and $D_2 = \{(0, 1)\}$. Clearly, the users' initial endowments of $(\frac{1}{2}, \frac{1}{2})$ are not Pareto optimal, in the sense that both users are better off if the entire first resource is allocated to user 1 and the entire second resource is allocated to user 2. Much research in game theory and economics studies how different strategic agents should cooperate when non-cooperation leads to Pareto suboptimal results. The Nash Bargaining (NB) mechanism implements one solution to this problem, given by John Nash [6].

Intuitively, NB allocates resources so as to maximize the product of all users' utilities. Consider, for instance, a resource pool of $(8, 8)$ and two users with demands $D_1 = \{(5, 1), (2, 2)\}$ and $D_2 = \{(1, 4), (2, 2)\}$. The allocation that maximizes the product of utilities is allocating $(4, 4)$ to each user, as in this scenario each user has a utility of 2 (as two tasks can be executed by each user) and the product gives $2 \cdot 2 = 4$ (and it can be verified that no other allocation leads to a higher value). Formally, NB outputs the allocation $Y = (Y_1, ..., Y_n)$ such that

$$\max \prod_{j \in N} u_j(Y_j) \tag{3}$$

$$\sum_{j \in N} Y_j \leq C \tag{4}$$

where $C = (C_1, \ldots, C_n)$.

IV. Results for Unrestricted Demands

We now explore the guarantees of the two mechanisms presented in Section III for general resource demands, i.e., when no restrictions whatsoever are imposed on users' resource demands. We consider the three main criteria presented in Section II: computational efficiency, fairness, and incentive compatibility. We then discuss the other desiderata presented in Section II: non-wastefulness and sharing incentive. The following table summarizes our results for general demands.

	CE	EF	SI	PO	LMMF	SP
LMMF	✓			✓	✓	
NB	✓	✓	✓	✓		

(CE = computationally efficient, EF = envy free, SI = sharing incentive, PO = Pareto optimal, LMMF = lexicographically max-min fair, SP = strategyproof)

A. Computational Efficiency

LMMF. Simple arguments (omitted) show that LMMF (which repeatedly solves linear programs) is computationally efficient

Proposition 1. *LMMF is computationally efficient*

NB. Recall that NB computes the allocation that maximizes the product of user's utilities. We show (proof is deferred to the full version) that this can be formulated as convex optimization and is thus computationally efficient.

Proposition 2. *NB is computationally efficient*

B. Fairness

LMMF. We first analyze the fairness properties of LMMF. The first step is proving that LMMF indeed computes a lexicographically max-min fair allocation. We then observe that LMMF is Pareto optimal and prove that it is not, however, envy free.

The nontrivial proof that the LMMF mechanism, as defined in Section III, indeed computes a lexicographically max-min fair allocation, follows from a combination of lemmas.

Claim 1. *For any two feasible allocations Y, Y' and $\gamma \in [0,1]$, the allocation $\hat{Y} = (\gamma Y_j + (1-\gamma)Y'_j)$ is feasible.*

We now claim that the function $u_j(\gamma Y_j + (1-\gamma)Y'_j)$ is concave in $\gamma \in [0,1]$ (formal proof is deferred to the full version). The intuition behind is that if j obtains the allocation $\gamma Y_j + (1-\gamma)Y'_j$ then it can always gain the utility of $\gamma u_j(X) + (1-\gamma)u_j(X')$ by using the support demand vectors (the vectors that are used for computing the utility) of $u_j(Y_j)$ and $u_j(Y'_j)$ in the right proportions.

Lemma 1. *For every $j \in N$, two resource vectors X and X', and $\gamma \in [0,1]$, $u_j(\gamma X_j + (1-\gamma)X'_j) \geq \gamma u_j(X) + (1-\gamma)u_j(X')$.*

We also give the following lemma (proof is deferred to the full version)

Lemma 2. *For every $j \in N$ and two resource vectors X and X', the function $f_j(\gamma) = u_j(\gamma X + (1-\gamma)X')$ is continuous for all $\gamma \in [0,1]$.*

We now introduce the notion of a "*pivot* user".

Definition 2. Given two allocation Y and Y', a pivot user of allocation Y with respect to Y' is the user with the lowest utility in Y that has different utility in Y'.

To illustrate that consider two allocations of resources to 4 users, Y and Y' such that $\langle Y \rangle = \langle 2,3,5,7 \rangle$ and $\langle Y' \rangle = \langle 1,3,5,6 \rangle$. Suppose that in $\langle Y \rangle$ coordinates 1-4 correspond to the utilities of users i_1-i_4, respectively, whereas in $\langle Y \rangle$ coordinate 1-4 correspond to users i_4, i_2, i_3, and i_1, respectively. Observe that in this scenario, the pivot of Y with respect to Y' is user i_1, whereas the pivot of Y' with respect to Y is user i_4 (see Figure 4). In fact, there can be more than a single pivot user, e.g., if, in the previous example, the value in the second coordinate of $\langle Y \rangle$ was changed from 3 to 2, then both user i_1 and user i_2 would fit the definition of a pivot user of Y with respect to Y'. Let $\theta(Y, Y')$ denote the set of all pivot users of allocation Y with respect to Y'.

Claim 2. *Let $i \in \theta(Y, Y')$ and $j \in \theta(Y', Y)$ be two pivot users. Then, $\langle Y' \rangle \succ \langle Y \rangle$ if and only if $u_j(Y'_j) > u_i(Y_i)$.*

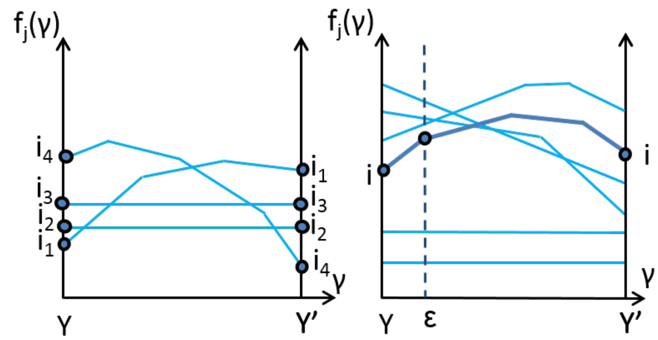

Fig. 4. The two figures describes the utilities of users $j \in N$ under allocations Y_j, Y'_j, and under the convex combination $f_j(\gamma)$ for $\gamma \in [0,1]$. Lemmas 1,2 state that $f_j(\gamma)$ is continuous and concave. In the left figure there are users i_1, i_2, i_3, i_4. The pivot users are $\theta(Y, Y') = \{i_1\}$ and $\theta(Y', Y) = \{i_4\}$. The right figure illustrates the statement of Corollary 1: a sufficient condition for allocation to not be LMMF is for a pivot user $i \in \theta(Y, Y')$ to have a higher utility in Y' than in Y. This is because there exists another allocation $\hat{Y} = \epsilon Y' + (1-\epsilon)Y$, for a sufficiently small $\epsilon > 0$, in which i is a pivot user (i.e., $i \in \theta(\hat{Y}, Y)$) such that $f_i(\epsilon) > u_i(Y)$ (Lemma 3). Hence, by Claim 2, $\hat{Y} \succ Y$. Thus, Y is not LMMF.

Lemma 3. *For any two allocations Y, Y' and pivot user $i \in \theta(Y, Y')$, if $u_i(Y'_i) > u_i(Y_i)$, then there exists another allocation $\hat{Y} = (1-\epsilon)Y + \epsilon Y'$ for a very small $\epsilon > 0$ such that $u_i(\hat{Y}_i) > u_i(Y_i)$.*

To prove this we use the concavity of the utility function and the continuity of the function $f_j(\gamma)$ in Lemma 2. (for intuition see Figure 4).

Combining Lemmas 2 and 3 gives the following:

Corollary 1. *For any two allocations Y, Y' and pivot user $i \in \theta(Y, Y')$, if $u_i(Y'_i) > u_i(Y_i)$, then Y is not LMMF.*

We are now finally ready to prove the following statement.

Proposition 3. *The LMMF mechanism is lexicographically max-min fair.*

Proof. Let Y be the allocation that the mechanism returns for a given input and let Y^* be a LMMF allocation. We show this by induction on the iteration number t. Note, that if by the end iteration t (see Section III) the mechanism created sets $p_1, ..., p_t$ (as described in Section III), and all users in these sets have the same utility in both Y and Y^*, then at iteration $t+1$ all users in $N \setminus P$, where $P = \bigcup_{i \in [t]} p_i$ can achieve the same minimal utility under Y and Y^* as the linear program that LMMF solves at the (t+1)'th iteration. Hence, in particular, the pivot in both allocations has the same utility. Suppose, for point of contradiction, that Y is not LMMF, and let t be the iteration at which the pivot $q = \theta(Y, Y^*)$ is added to P. Then, by definition of the pivot q, it must be that $u_q(Y^*) > u_q(Y)$. Thus, by Lemma 3, there is an allocation $\hat{Y} = \epsilon Y^* + (1-\epsilon)Y$ such that $u_q(Y) < u_q(\hat{Y})$. However, this contradicts the fact that the linear program of the LMMF mechanism maximizes the utility of all users in $N \backslash P$ at iteration t. ☐

We next show that LMMF is Pareto optimal (simple proof deferred to the full version) but is not envy free.

Claim 3. *LMMF is Pareto optimal.*

Proposition 4. *LMMF violates envy freeness.*

Proof. Consider the following example:

Example 3. A resource pool of a single resource $C = (1)$ and two users, each with single demand vector: $D_1 = \{(1)\}$ and $D_2 = \{(0.5)\}$.

The allocation under LMMF is $Y_1 = \frac{2}{3}$ and $Y_2 = \frac{1}{3}$ as both users get a utility of $\frac{2}{3}$. In this scenario, user 2 prefers the allocation of user 1 over his own, and hence LMMF is not EF. ☐

NB. We now prove that NB is both envy free and Pareto optimal.

Proposition 5. *NB is envy free and Pareto optimal.*

Proof. The Competitive Equilibrium from Equal Income (CEEI) [7] in economic theory is the allocation reached in a competitive market with multiple resources when each agent starts with an "endowment" of $\frac{1}{n}$ of each resource and then trades resources with the others. To prove the proposition we show that the allocation of NB coincides with that of the CEEI. To establish this, it suffices to show that the utility function of each user j in our setting is homogeneous, in the sense that $\gamma u_j(X) = u_j(\gamma X)$ for every scalar γ and vector of resources $X = (X_1, ..., X_k)$ (see [6], [7]). To see this, note that by the definition of a utility function in our model

$$u_j(\gamma X) = \max \sum_{m=1}^{M_j} \alpha_m \quad (5)$$

$$\text{Subject to} \quad \sum_{m=1}^{M_j} \alpha_m d_{jm} \leq \gamma X \quad (6)$$

$$\alpha_m \geq 0 \quad \forall m \in [M_j]$$

Constraints 6 can be rewritten as $\sum_{m=1}^{M} \frac{\alpha_m d_{jm}}{\gamma} \leq X$. Hence we can define $\alpha'_m \triangleq \frac{\alpha_m}{\gamma}$ and constraint (6) can be written as $\sum_{m=1}^{M} \alpha'_m d_{jm} \leq X$. Since $\max \sum_{m=1}^{M} \alpha_m = \gamma \max \sum_{m=1}^{M} \alpha'_m$ we get $\gamma u_j(X) = u_j(\gamma X)$. Hence, the allocation returned by NB is equivalent to the outcome of CEEI. CEEI is known to be PO and EF [8] and so the proof immediately follows. ☐

We point out that NB can easily be seen to not be lexicographically max-min fair. In fact, even if there is a single resource, NB will always split that resource equally disregarding how much it is worth to each user as the utility function is homogeneous.

C. Incentives

Proposition 6. *LMMF is not strategyproof.*

Proof. Consider Example 3. User 2 benefits by reporting $D'_2 = \{(1)\}$, since then the resource is divided equally. ☐

Proposition 7. *NB is not strategyproof.*

Proof. Consider the following example: A resource pool of $C = (1,1)$ and two users with single demand vector each" $D_1 = \{(\frac{2}{3}, \frac{1}{3})\}$ and $D_2 = \{(\frac{1}{4}, \frac{3}{4})\}$. NB outputs $Y_1 = (\frac{12}{15}, \frac{6}{15}), Y_2 = (\frac{3}{15}, \frac{9}{15})$ and the corresponding utilities are $u_1(Y_1) = 1.2$ and $u_2(Y_2) = 0.8$. If user 2 reports $D'_2 = \{(\frac{1}{3}, \frac{2}{3})\}$ then NB outputs $Y_1 = (\frac{2}{3}, \frac{1}{3}), Y_2 = (\frac{1}{3}, \frac{2}{3})$ and the corresponding utilities are $u_1(Y_1) = 1$ and $u_2(Y_2) = 0.83$ (note that since user 2 misreports, not all of his obtained resources are in use). Hence, by misreporting, user 2 can improve his utility from 0.8 to 0.83. ☐

Proposition 8. *LMMF violates sharing-incentive.*

Proof. Consider resource pool $C = (1,1,1)$ and two users with demands $D_1 = \{(0,1,0)\}, D_2 = \{(1,0,0), (0, \frac{1}{2}, \frac{1}{2})\}$, where each user has "endowment" $E = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. Note that $u_2(E) = \frac{3}{2}$. Then, the allocation under LMMF are given by $Y_1 = (0,1,0), Y_2 = (1,0,0)$. Hence, $u_2(Y_2) = \frac{1}{2}$ and then $u_2(E) > u_2(Y_2)$. ☐

Proposition 9. *NB satisfies sharing-incentive.*

Proof. As shown in Lemma 5, the output of NB is equivalent to CEEI. As CEEI satisfies sharing incentive, the proof follows. ☐

V. Results for Linear Tradeoffs

As seen in the previous section, the LMMF mechanism does not guarantee many desired properties. We now show that in the very restricted setting in which tradeoffs are linear, that is, $\sum_r d^r_{jm} = 1$ for each user $j \in N$ and demand $d_{jm} \in D_j$, LMMF is, in fact, both envy free (EF) and strategyproof (SP). The following table summarizes our results for LMMF and NB for this class of so called "simplex demands". We regard these results as a first step and leave the analysis of other restricted classes of demands of interested to future research.

Properites	CE	EF	SI	PO	LMMF	SP	GSP
LMMF	✓	✓		✓	✓	✓	✓
NB	✓	✓	✓	✓			

We next present our proofs that LMMF is both EF and SP. The following lemma plays an important role in our proofs.

Lemma 4. *Let D be a specification of all users' resource-demands and u be the utility function over demands in D. Similarly, let D' be a different specification of all users' resource-demands, and u' be a utility function over demands in D'. Let $Y = (Y_1, \dots, Y_n)$ be the output of the LMMF mechanism for D. Then, $u_i(Y_i) \geq u'_i(Y_i)$.*

Proof. Since the allocation Y_i is a convex combination of the resource demand vectors, the "packing coefficients" of u_i, i.e., $\alpha_1, ..., \alpha_{M_i}$, satisfies

$$\sum_{m=1}^{M_i} \alpha_m d^r_{i_m} = Y^r_i, \quad r \in [k] \tag{7}$$

Since D' is an arbitrary demand set, Y_i is not necessarily a combination of demand vector in D'_i. Thus, the "packing coefficients" of u', i.e., $\alpha'_1, ..., \alpha'_{M'_i}$, satisfies,

$$\sum_{m=1}^{M'_i} \alpha'_m d'^r_{i_m} \leq Y^r_i, \quad \forall r \in [k] \tag{8}$$

Summing the constraints (7) and (8) over all resources,

$$\sum_{r\in[k]} \sum_{m=1}^{M'_i} \alpha'_m d'^r_{i_m} \leq \sum_{r\in[k]} \sum_{m=1}^{M_i} \alpha_m d^r_{i_m} \tag{9}$$

Changing the order of summation and using the property that the demand vectors are over the simplex,

$$\sum_{m=1}^{M'_i} \alpha'_m \leq \sum_{m=1}^{M_i} \alpha_m$$

By definition of utility , $u_i(Y_i) \geq u'_i(Y_i)$. □

We are now ready to prove SP.

Theorem 1. *LMMF mechanism over simplex demands is strategyproof.*

Proof. Let i be a manipulative user and suppose by contradiction that i benefits by misreporting D'_i instead of reporting its real demand D_i. Let $D' = (D'_i, D_{-i})$ be the report of all users under the manipulation of i, and let Y' be the resulting allocation of LMMF under D'. Let Y be the resulting allocation of LMMF given that all users report their true demands (under D). We first show that if i benefits by lying then $i \in \theta(Y, Y')$; namely i is a pivot user. Let $q \in \theta(Y, Y')$ and $q' \in \theta(Y', Y)$ be pivot users. Suppose, for point of contradiction, that $u_q(Y_q) \leq u_i(Y_i)$. We now analyze three possibilities and show that under at each, there exists an allocation that contradicts the property of LMMF (see figure 5):

1) $u_q(Y_q) < u_{q'}(Y'_{q'})$. Then, followed by Claim 2, $\langle Y' \rangle \succ \langle Y \rangle$.
2) $u_q(Y_q) = u_{q'}(Y'_{q'})$. First note that it must be that $q \neq q'$. as otherwise it contradict the fact that q and q' are pivots. We can also derive by the definitions of the pivots q and q' that $u_q(Y'_q) \geq u_{q'}(Y_{q'})$. Since $u_q(Y_q) = u_{q'}(Y'_{q'})$ we get $u_q(Y'_q) \geq u_q(Y_q)$. Followed by corollary 1, we get a contradiction to LMMF.
3) $u_q(Y_q) > u_{q'}(Y'_{q'})$. To show a contradiction to LMMF, we define a dual scenario where the demand set of the users is D' instead of D, i.e., i's real demand is D'_i and the manipulation is D_i. Thus, in this case Y' is the truthful allocation (under D'). Let u' be the dual utility function, namely u' is utility function over demand D'. Since all users except for i report their true demands, their utility under the same allocation over demand sets D and D' is the same. Formally, for each user $j \neq i$

$$u'_j(Y'_j) = u_j(Y'_j) \text{ and } u'_j(Y_j) = u_j(Y_j) \tag{10}$$

Thus, we get a 'mirror image' of the utilities of the users, except for i. Note that without the presence of i, we had $q' \in \theta'(Y', Y)$, and hence, $u'_{q'}(Y_{q'}) > u'_{q'}(Y'_{q'})$, which by corollary 1, leads to a contradiction that Y' is not LMMF over D'. We need to show that q' is the pivot (with the presence of i). Indeed, $q' \in \theta'(Y', Y)$ since $u'_i(Y') > u'_{q'}Y'$, which followed by,

$$u'_i(Y') \geq u_i(Y') > u_i(Y) > u_q(Y) > u_{q'}(Y) = u'_{q'}(Y')$$

where the left inequality follows from Lemma 4, the second left inequality uses the assumption of manipulator, the third left inequality uses the case condition, the second right inequality follows as i is not a pivot, and the right equality follows (10) (this can be pictures in case 3 of figure 5).

Thus we get that the manipulative user i has to be the pivot (i.e, $i \in \theta(Y, Y')$). By assumption of the manipulative user $u_i(Y'_i) > u_i(Y_i)$. But then, by corollary 1, Y is not LMMF. Thus, if i benefits by misreporting its demand, then this leads a contradiction to LMMF. Hence, LMMF mechanism over simplex demands is strategyproof. □

We note here that we can strengthen Theorem 1, showing that LMMF under simplex demand is *Group-strategyProof* (GSP). This can be proved using very similar argument to the proof of Theorem 1, showing that if there exists some manipulator user that improve its utility by misreporting, there must exists another manipulator user that is worse off as it misreports its demand vectors.

Theorem 2. *LMMF is envy-free over simplex demands.*

Proof. Let Y be an LMMF allocation. Suppose, by contradiction, that LMMF mechanism is not envy-free. This implies that there is a pair of user i, j such that $u_i(Y_j) > u_i(Y_i)$.There are two complementary cases:

1) Suppose that $u_i(Y_i) \geq u_j(Y_j)$. By Lemma 4 we get $u_j(Y_j) \geq u_i(Y_j)$ (as i's demand set is different from

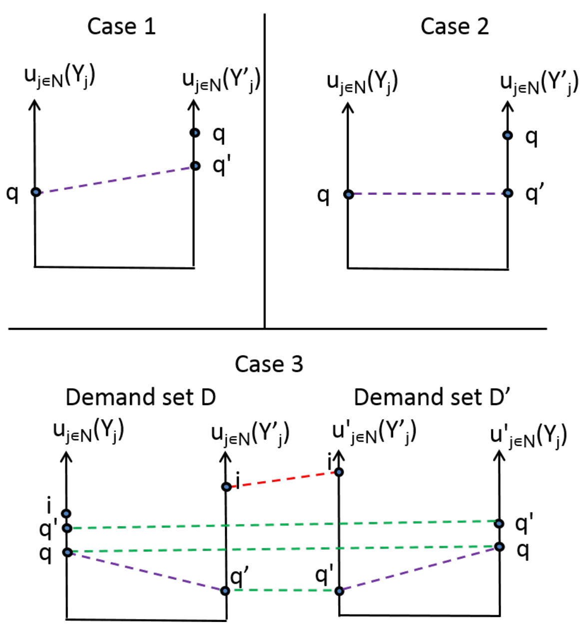

Fig. 5. Illustration of the cases in the proof of Theorem 1: in each case there is a pivot user (q in cases 1 and 2, and q' in case 3) that has higher utility in another allocation (which implies that Y is not LMMF under demand set D, or Y' is not LMMF under demand set D'). Case 3 analyzes the dual scenario under demand set D' (which is a mirror image of the original case, as shown by the green dashed line, except for user i). By Lemma 4 we have $u'_i(Y') \geq u_i(Y')$ (as shown by the red dashed line).

j's). Therefore, we have $u_i(Y_i) \geq u_i(Y_j)$. Contradiction to the assumption.

2) Suppose that $u_i(Y_i) < u_j(Y_j)$. In this case we argue that $u_i(Y_j) = 0$. Suppose by contradiction that $u_i(Y_j) > 0$. Let $\delta > 0$ be a (small) constant to be determined later. Suppose that user j transfers the vector $\delta \cdot Y_j$ to user i. Then the utility of j is $u_j(Y_j - \delta Y_j)$ and the utility of user i becomes $u_i(Y_i + \delta Y_j)$. Since the utility function is continuous (Lemma 2), for sufficiently small $\delta > 0$ it holds that $u_j(Y_j - \delta Y_j) > u_i(Y_i)$. Since the utility function is concave (Lemma 1), it follows that $u_j(\delta Y_j) \geq \delta u_j(Y_j)$. Using the assumption that $u_i(Y_j) > 0$ it follows that $u_i(Y_i + \delta Y_j) > u_i(Y_i)$. Thus, the vector $(u_i(Y_i + \delta Y_j), u_j(Y_j - \delta Y_j))$ is lexicographically larger than $(u_i(Y_i), u_j(Y_j))$. Let Y' be the allocation after the transfer. Since $Y_l = Y'_l$ for every $l \neq i, j$), we get $\langle Y' \rangle \succ \langle Y \rangle$. This is a contradiction to LMMF. Hence, $u_i(Y_j) = 0$. But then this implies that $u_i(Y_j) \leq u_i(Y_i)$ (as the utilities are non negative) and therefore contradicts the assumption.

□

LMMF is not SI under simplex demands, as the counter example in section IV considers users with simplex demands.

VI. Extensions

We now briefly discuss extensions of the results presented in previous sections. Recall that the utility function considered in our model (Section II) reflects the number of resource-demands that can be packed in a given set of resources. Our results for LMMF actually extend to utility functions of the form

$$u_j(X) = \max_{\alpha_1,\alpha_2,\ldots,\alpha_{M_j}} f\Big(\sum_{m=1}^{M_j} \alpha_m d_{jm}\Big)$$
$$s.t. \quad \sum_{m=1}^{M_j} \alpha_m d_{jm} \leq X$$
$$\alpha_m \geq 0 \quad \forall m \in [M_j]$$

where f is a concave function and satisfies strict monotonicity (i.e., for any positive γ, $f((1+\gamma)x) > f(x)$). In fact, our results for LMMF hold even if each user has a different function f, so long as these functions are common knowledge. Lastly, when f is strictly concave, the LMMF outcome is guaranteed to be unique.

VII. Related Work

Fairly allocating resources to different parties is an age-old challenge and a prominent research area in game theory and economics. More recently, fair allocation of resources has also received much attention from a computer science perspective. Much of the study of fair allocation in computer science focused on the allocation of a single resource (e.g., the famous "cake cutting" setting [9], [10]). Also, much work on fair allocation of multiple resources deals with multiple units of the same resource, e.g, fair scheduling [11], [12], [13], and fairly allocating link bandwidth [14], [15], [16]. Fair scheduling of multiple resources was implemented for Dryad [17] and Hadoop [18]. However, although dealing with heterogeneous resource demands, these mechanisms abstract the multiple resources into a single resource, and consquently sometimes inefficiently utilize resources.

Ghodsi et al. [1] provided a general framework for resource allocation with heterogeneous resource demands. [1] presents the so called "Dominant Resource Fairness" (DRF) mechanism, which equalizes the dominant resource (i.e., the most demanded resource) each user receives, and prove that DRF satisfies strategyproofness, envy-freeness, Pareto optimality, and sharing incentive. Parkes et al. [5] later proved that DRF actually satisfies group strategyproofness—a stronger notion of incentive compatibility. Dolev et al. [19] proposed a different notion of fairness for heterogeneous resources called "no justified complaints" and proved the guaranteed existence of a fair allocation in this sense. Gutman and Nisan [20] generalized the notion of DRF and introduced polynomial-time algorithms for computing fair allocations for both this generalized notion and the orthogonal notion the "no justified complaints". Recently, Wang et al. [21] proposed DRFH, a generalization of DRF to an environment with multiple heterogeneous servers, and showed that it satisfies Pareto optimality, envy freeness, strategyproofness, as well as other interesting properties.

Lan et al. presented an axiomatic approach to measuring fairness in a single-resource domain[22]. This was generalized in [2] to heterogeneous resources by considering two measures of fairness, GFJ and FDS, where GFJ measure fairness in term of number of jobs allocated to each user and FDS measure fairness in terms of the relative size of the dominant share. [2] presents conditions on these measures to achieve Pareto optimality, sharing incentive and envy freeness.

Two notions of fairness that play the key roles in our work are (lexicographic) max-min fairness and proportional fairness. Max-min fairness is a classical notion that dates back to Rawls [23]. Algorithms that implement max-min fairness include various round robin schemes, proportional resource sharing [24], weighted fair queuing [25] and bandwidth allocation [26]. We note that alternative notions of fair allocation were introduces. Other notions of fairness have also been considered in networking contexts, e.g., Foster et al. [27] proposed several fairness criteria for network allocation such as min guarantee (guaranteeing minimal bandwidth for every virtual machine), and payment proportionality (where bandwidth allocation is based on payments) and presented different mechanisms. Nash Bargaining was introduced by John Nash [6] as a general approach to collaboration in environments with self-interested parties that satisfies several axioms, including Pareto optimality. Nash bargaining coincides with the well-studied notion of proportional fairness and was shown to coincide with market equilibria for a large class of utility functions [8], [7]. Cole, et al. proposed a truthful mechanism that approximates the Nash Bargaining solution [28], but at the cost of "throwing away" a large fraction of the resources.

VIII. Conclusion

We initiated the study of fair resource-allocation with resource tradeoffs. We leave the reader with two interesting research directions: (1) We proposed and theoretically analyzed two mechanisms: LMMF and NB. Examining other mechanisms and, in particular, mechanisms that reflect other economic approaches to fair resource allocation is an interesting research direction; (2) We considered two opposite environments, namely, unrestricted tradeoffs and linear tradeoffs. Exploring other resource tradeoffs scenarios on this spectrum is another interesting agenda. In particular, empirical studies of actual resource tradeoffs in cloud computing might motivate new questions along the lines outlined in this paper.

Acknowledgements

David Hay and Michael Schapira are supported by the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11). David Hay is also supported by the Neptune Consortium, administered by the Office of the Chief Scientist of the Israeli ministry of Industry and Trade and Labor.

References

[1] 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 NSDI*, vol. 11, 2011, pp. 24–24.

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

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. The MIT Press, 2009.

[4] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks."

[5] D. C. Parkes, A. D. Procaccia, and N. Shah, "Beyond dominant resource fairness: extensions, limitations, and indivisibilities," in *Proc. ACM EC*, 2012, pp. 808–825.

[6] J. Nash, "The bargaining problem," *Econometrica: Journal of the Econometric Society*, 1950.

[7] H. Moulin, *Fair division and collective welfare*. MIT press, 2004.

[8] H. Varian, "Equity, envy, and efficiency," *Journal of economic theory*, 1974.

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

[10] J. B. Barbanel, A. D. Taylor *et al.*, *The geometry of efficient fair division*. Cambridge University Press Cambridge, 2005.

[11] S. K. Baruah, J. E. Gehrke, and C. G. Plaxton, "Fast scheduling of periodic tasks on multiple resources," in *Proc. IEEE PPS*, 1995, pp. 280–280.

[12] D. Zhu, D. Mossé, and R. Melhem, "Multiple-resource periodic scheduling problem: how much fairness is necessary?" in *Proc. IEEE RTSS*, 2003, pp. 142–151.

[13] Y. Liu and E. Knightly, "Opportunistic fair scheduling over multiple wireless channels," in *Proc. IEEE INFOCOM*, vol. 2, 2003, pp. 1106–1115.

[14] J. Kleinberg, Y. Rabani, and É. Tardos, "Fairness in routing and load balancing," in *Proc. IEEE FOCS*, 1999, pp. 568–578.

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

[16] F. P. Kelly, A. K. Maulloo, and D. K. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research society*, pp. 237–252, 1998.

[17] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: fair scheduling for distributed computing clusters," in *Proc. ACM SIGOPS*, 2009, pp. 261–276.

[18] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in *Proc. ACM EuroSys*, 2010, pp. 265–278.

[19] 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 ITCS*, 2012, pp. 68–75.

[20] A. Gutman and N. Nisan, "Fair allocation without trade," in *Proc. AAMAS*, 2012, pp. 719–728.

[21] W. Wang, B. Li, and B. Liang, "Dominant resource fairness in cloud computing systems with heterogeneous servers," *arXiv preprint arXiv:1308.0083*, 2013.

[22] T. Lan, D. Kao, M. Chiang, and A. Sabharwal, *An axiomatic theory of fairness in network resource allocation*. IEEE, 2010.

[23] J. Rawls, "Some reasons for the maximin criterion," *The American Economic Review*, 1974.

[24] C. A. Waldspurger and W. E. Weihl, "Lottery scheduling: Flexible proportional-share resource management," in *Proc. USENIX OSDI*, 1994, p. 1.

[25] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. ACM SIGCOMM*, vol. 19, no. 4, 1989, pp. 1–12.

[26] B. Radunović and J.-Y. L. Boudec, "A unified framework for max-min and min-max fairness with applications," *IEEE/ACM Transactions on Networking (TON)*, vol. 15, no. 5, pp. 1073–1083, 2007.

[27] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Proc. IEEE GCE*, 2008, pp. 1–10.

[28] R. Cole, V. Gkatzelis, and G. Goel, "Mechanism design for fair division: allocating divisible items without payments," in *Proc. ACM EC*, 2013, pp. 251–268.