IEOR8100: Economics, AI, and Optimization
Lecture Note 10: Strategyproof Fair Division and Dominant Resource Fairness

Christian Kroer[*]

March 11, 2020

1 Introduction

In this lecture note we start the study of *strategyproofness.* So far in fair division, we did not worry about the following crucial aspect: we might not necessarily know the utility function u_i of each agent. Instead, we would often need to elicit u_i from agent i. In that case, we need to worry about whether each agent will actually tell us their true u_i. If they can tell us some other u'_i which leads to a better outcome for agent i, then we should not expect them to tell us u_i. This can lead to several serious issues: first, we cannot expect to maximize an objective that depends on u_i if we optimize for the wrong u'_i. Second, the market can be extremely volatile if agents are constantly shifting their behavior in order to best misreport based on what everybody else is doing. Third, it disadvantages agents that for one reason or another cannot strategize as effectively.

2 Setting and Strategyproofness

As before, we study fair division problems with the following setup: we have a set of m infinitely-divisible goods that we wish to divide among n agents. Without loss of generality we may assume that each good has supply 1. We will denote the bundle of goods given to agent i as x_i, where x_{ij} is the amount of good j that is allocated to agent i. Each agent has some utility function $u_i(x_i) \in \mathbb{R}_+$ denoting how much they like the bundle x_i. We shall use x to denote an assignment of goods to agents.

As a running example in this note we will use the setting of sharing compute resources.

Definition 1 (Compute-resource setting)**.** *In this setting we assume that each user has a specific type of task that they wish to run on a compute cluster. Each user's utility is linear in the (fractional) amount of tasks they complete. A task is parameterized by a vector $\{d_{ij}\}_j$, where $d_{ij} > 0$ is the amount of good j (in this setting goods are also called* resources*) that agent i needs in order to run one unit of their task. It is assumed without loss of generality that $d_{ij} \leq 1$ for all i, j, and that for all i there exists some j such that $d_{ij} = 1$, we call this the* dominant resource *for user i (if there is more than one such j then we pick one arbitrarily and label it the dominant resource). This*

[*]Department of Industrial Engineering and Operations Research, Columbia University. Email: christian.kroer@columbia.edu.

setting is equivalent to the Leontief utilities that were previously discussed under the Eisenberg-Gale convex program. Formally, an agent's utility ends up being:

$$u_i(x_i) = \min_j \frac{x_{ij}}{d_{ij}},$$

which is simply the number of tasks they are able to run.

As an example, consider an agent that needs 2 units of CPU and 1 unit of RAM in order to run their task (this would be represented as $\left(1, \frac{1}{2}\right)$). This agent would be indifferent between the allocations $(4, 2)$ and $(5, 2)$, since they are RAM-constrained on both allocations and can do nothing with the additional unit of CPU.

We now ask what may happen when agents can misreport their preferences. As a first example, we will consider two real-world mechanisms that are easily gamed. A popular approach in scheduling systems is to view resources simply as fixed *slots*, where a slot comes with a fixed amount of CPU, RAM, etc. Allegedly, a Yahoo! Hadoop MapReduce datacenter was designed with a fixed number of two slot types: map slots and reduce slots (*map* methods typically do things like filtering or sorting data, while *reduce* operations perform summary operations). A user discovered that map slows were much more constrained than reduce slots, and exploited this by writing very heavy reduce methods that also performed the work that would typically be done in the map method.

We now give an example showing that the CEEI mechanism is not strategyproof for the compute-resource setting.

Example 1. *Consider a compute-resource setting with two resources. User 1 has weights* $\left(1, \frac{1}{16}\right)$ *and user 2 has weights* $\left(\frac{1}{2}, 1\right)$*. The CEEI allocation allows user 1 to solve* $\frac{16}{31}$ *tasks and user 2 to solve* $\frac{30}{31}$ *tasks. If user 1 instead reports weights* $\left(1, \frac{1}{2}\right)$ *then the assignment is* $\frac{2}{3}$ *tasks for each user, thus increasing the utility of user 1.*

Even outside Leontief utilities, CEEI and market equilibria can in general be manipulated when agents report their utility functions. From the market equilibrium perspective, one way to think of it is that by misreporting, agents may change prices on certain goods in a way that favors them.

3 Dominant Resource Fairness Mechanism

In the compute-resource setting, it turns out that there is a fair division approach that achieves all the properties that we previously obtained for CEEI, while maintaining strategyproofness. Just to recall, remember that we wanted Pareto optimality (every other allocation makes at least one agent worse off, unless all utilities are the same), no envy (each agent prefers their own bundle to that of any other agents), and proportionality (each agent is at least as happy as they would be when assigned $\frac{1}{n}$ of each good).

The *dominant resource fairness* (DRF) mechanism allocates to each agent proportional to their weight vector d_i times a scalar $\alpha > 0$ that is chosen such that the allocation becomes feasible. The allocation to agent i is thus $x_i = \alpha d_i$. The scalar α is chosen as

$$\alpha = \frac{1}{\max_j \sum_i d_{ij}},$$

which is the largest possible value for maintaining feasibility, given our choice of allocation proportional to d_i.

It turns out that in the compute-resource setting DRF satisfies all the criteria we specified.

Theorem 1. *DRF satisfies Pareto optimality, no envy, proportionality, and strategyproofness for Leontief utilities where $d_{ij} > 0$ for all i, j.*

Proof. Let x be the DRF allocation.

Pareto optimality: in order to keep everybody at least as happy as under x, any alternative allocation x' must satisfy $x'_{ij} \geq x_{ij}$ for all i, j. But this means that only goods not allocated under x can be used to try to improve, and by our choice of α and the fact that $d_{ij} > 0$ for all i, j, we have that some item is constraining *every* agent's utility.

No envy: Consider a pair of agents i, i'. Now consider the dominant resource j such that $d_{ij} = 1$. We have that $d_{ij} = 1 \geq d_{i'j}$, which implies that $x_{ij} = \alpha d_{ij} \geq \alpha d_{i'j} = x_{i'j}$, and thus due to the Leontief utilities, i likes their own bundle at least as well as that of i'.

Proportionality: We have $\alpha \geq \frac{1}{n}$, since choosing $\frac{1}{n}$ is guaranteed to be feasible, as $d_{ij} \leq 1$ for all i, j. Thus every agent gets at least as much of their dominant resource under αd_i as under the $\frac{1}{n}$, and for the proportional allocation their dominant resource is their constraining item.

Strategyproofness: Consider any report d_{-i} of weights for other agents. Now consider the truthful report d_i and any misreport d'_i, with associated allocations x, x'. First consider the case where $\alpha' \leq \alpha$. In that case, agent i gets less of their dominant resource and so their utility does not improve. Second, if $\alpha' > \alpha$ then consider the item j such that $\sum_i x_{ij} = 1$. Under α' agent i must get strictly less of i, since every other agent consumes strictly more of j, and thus their utility decreases. $\square$

Thus, DRF for the compute-resource setting satisfies all the desiderata we stipulated. However, in practice we won't know all of the items and goods up front. Instead, users would typically arrive over time and we would only learn the utility function of a given agent when they submit their tasks. We now consider that setting.

4 Dynamic DRF

Formally, we consider a model where the number of goods m is known in advance, and the total number of users n is known. However, each user i does not reveal their utility function weights d_i until they arrive (we still assume $d_{ij} > 0$ for all i, j). The users arrive in index order $1, 2, \ldots, n$, and at time k we have k users in the system, and we must perform an allocation x^k of the m goods to the k users currently present. Crucially, the dynamic DRF setting assumes that resources are *irrevocable*, and so $x^{k+1}_{ij} \geq x^k_{ij}$ is a constraint on the allocation at time $k + 1$.

A natural goal in dynamic DRF is to extend the four properties satisfied by DRF to the dynamic setting. The properties and their dynamic counterparts are listed in the table below

Property	Static (DRF)	Dynamic
Envy free	No agent wishes to swap with another agent: $u_i(x_i) \geq u_i(x_{i'}),\ \forall i, i'$	No agent present at time k wishes to swap with any other agent present at time k: $u_i(x_i^k) \geq u_i(x_{i'}^k),\ \forall i, i' \leq k$.
Proportionality	x_i is at least as good as the proportional $\frac{1}{n}$ allocation: $u_i(x_i) \geq u_i(\vec{1}\frac{1}{n}),\ \forall i$	At every timestep k, we satisfy proportionality: $u_i(x_i^k) \geq u_i(\vec{1}\frac{1}{n}),\ \forall i \leq k$
Strategyproofness	No gains by misreporting	No gains by misreporting at any time step
Pareto optimality	No Pareto-improving allocation	No Pareto-improving allocation that uses at most $\frac{k}{n}$ of every resource

It turns out that the above desiderata put strong constraints on the types of dynamic allocations we may consider.

First, in order to satisfy dynamic Pareto optimality (DPO) at time k, it is easy to see that at least one good must use at least $\frac{k}{n}$ of its supply. If this does not hold, some agent can be allocated an ϵ amount of every item, and thereby increase their utility.

Secondly, dynamic proportionality requires us to use *at most* $\frac{k}{n}$ of every good at time k. Say we used more than $\frac{k}{n}$ of some good j, then the remaining agents $k+1, \ldots, n$ could show up with j as their dominant resource, and we would not be able to maintain dynamic proportionality for those agents.

We get from the above two paragraphs that our allocation mechanism must be such that the item (or items) with the highest amount allocated at time k must have exactly $\frac{k}{n}$ of their supply allocated.

However, it turns out that envy freeness is incompatible with DPO.

Theorem 2. *Let $n \geq 3, m \geq 2$. Then no dynamic allocation mechanism satisfies envy freeness and DPO.*

Proof. Let $n = 3, m = 2$. Agents 1 and 2 have utility vectors $d_1 = (1, \frac{1}{9})$ and $d_2 = (\frac{1}{9}, 1)$ respectively. At timestep 2, either item 1 or 2 must have $\frac{2}{3}$ of their supply allocated in order to satisfy DPO. Then either agent 1 or 2 must be allocated at least $x = \frac{3}{5}$ of their dominant item, since $x + \frac{1}{9}x = \frac{2}{3}$. If this does not hold then we could Pareto improve by allocating more to either one of the agents. Say agent 1 gets at least $\frac{3}{5}$ of item 1 without loss of generality. Now when the third agent arrives, say they also have utility vector $d_3 = (1, \frac{1}{9})$. Then we can give them at most $\frac{2}{5}$ of item 1, and thus they envy agent 1.

The argument extends easily to more agents by adding duplicates of agent 3 and showing a similar envy condition. More items can be added simply by having each agent require a negligible amount of each additional item. $\square$

Thus it is unfortunately impossible to get every condition that we care about. Instead we will have to either drop or relax some of the conditions. If we were to drop DPO, there is a trivial mechanism satisfying the remaining desiderata: perform the proportional allocation to every agent k at time k. This clearly satisfies envy freeness, strategyproofness, and proportionality. Similarly, if we drop envy freeness, then the following *dynamic dictator* mechanism achieves DPO, strategyproofness, and proportionality: at time k, allocate to agent k $\frac{1}{n}d_i$. If $k > 1$, proceed to allocate items to agent 1 proportional to d_1 until some item has used $\frac{k}{n}$ of its supply. As an exercise, prove that this is strategyproof and DPO (it clearly satisfies proportionality).

Instead of completely dropping either envy freeness or DPO we can try to relax them. First, let us consider relaxing envy freeness. We know that there will generally be envy in our allocations. But at time k, if agent $i < k$ is envied by others, then we probably should not be allocating more goods towards agent i before fixing the envy. This motivates

Definition 2 (Dynamic envy freeness (DEF))**.** *If agent i envies agent i', then agent i' must have arrived before agent i, and i' must not have been allocated any resources since i arrived.*

It turns out that DEF, DPO, strategyproofness, and proportionality, is maintained by the following mechanism, called *dynamic DRF*: at time k, we have some previous allocation x^{k-1} that we must respect. Now continuously allocate towards *each* agent that has the minimum dominant share at the same rate, until a $\frac{k}{n}$ fraction of at least one good is allocated. By allocating in this fashion, we ensure that no agent which is currently envied ever gets any allocation. This "water-filling" algorithm is a natural dynamic extension of the DRF mechanism (where we increased *all* allocations at the same rate, proportional to the d_is).

The correct allocation at each time-step k can be computed via linear programming.

5 Historical Notes

Ghodsi et al. [1] introduced the concept of dominant-resource fairness, in the context of fairly dividing multiple resource types, specifically motivated by compute-cluster resource sharing. Most of the examples given here are from that paper. Parkes et al. [3] extend the original DRF paper by proving that DRF satisfies group strategyproofness and giving several other extensions as well as more formal proofs. Kash et al. [2] introduce the dynamic setting and provide the impossibility result described here, the dynamic DRF mechanism, as well as a mechanism for handling relaxed DPO.

References

[1] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Nsdi*, volume 11, pages 24–24, 2011.

[2] Ian Kash, Ariel D Procaccia, and Nisarg Shah. No agent left behind: Dynamic fair division of multiple resources. *Journal of Artificial Intelligence Research*, 51:579–603, 2014.

[3] David C Parkes, Ariel D Procaccia, and Nisarg Shah. Beyond dominant resource fairness: Extensions, limitations, and indivisibilities. *ACM Transactions on Economics and Computation (TEAC)*, 3(1):1–22, 2015.