Several Stories about High-Multiplicity EFx Allocation

Nikita Morozov,[1] Artur Ignatiev,[2] Yuriy Dementiev [2]

[1] Constructor University Bremen, Germany
[2] HSE University, St. Petersburg, Russia
nmorozov@jacobs-university.de, aaignatiev@hse.ru, ydementyev@hse.ru

Introduction

Fair and efficient allocation of resources is a very important issue in Economics and Computer Science. First mentioned in the mid-20th century, it arises in a variety of practical applications, such as dividing rewards among groups, allocating students to courses, and assigning tasks within a team. One of the most popular notion of fairness is envy-freeness (EF), which requires that every agent prefers their own bundle of goods to that of any other. However in the case of indivisible goods, EF allocations may not exist. This motivated the study of its relaxations. One of the actual and relevant relaxations of EF proposed by Caragiannis et al. [1], is called envy-free up to any item (EFx). Each agent's bundle should be worth at least as much as any other agent's bundle minus any single item for the allocation to be EFx. The existence of EFx allocations is considered as the biggest open question in fair division. We refer to overview paper [2] for a detailed overview of fair division of indivisible goods.
The standard notion of efficiency is Pareto optimality (PO). An allocation is said to be PO if no other one makes an agent better off without making someone else worse off. An important question in fair division is whether the notions of fairness can be achieved in conjunction with the efficiency notions PO. In general, EFx + PO allocations are not guaranteed to exist [3]. In this paper we focus on the algorithmic complexity of finding EFx + PO (EFx and at the same time PO allocation).

Setting

1. N is a set of n agents.
2. M is a set of m goods that cannot be divided or shared.
3. Each agent $i \in N$ is equipped with an additive valuation function $v_i : 2^M \to \mathbb{N}_{\geq 0}$, which assigns a non-negative integer and $v_i(S) = \sum_{g \in S} v_i(g)$ for any subset of items $S \subseteq M$.
4. Item type is a vector of length n, where the i-th coordinate is the value of the good's utility for the i-th agent. We will use k to denote the number of item types.

A fair allocation instance is denoted by $I = (N, M, v)$ where $v = (v_1, ..., v_n)$ is the vector of valuation functions and can be represented by a table with a row per agent and a column per good, such that cell (i, g) contains the value $v_i(g)$.
An allocation is a tuple of subsets of M : $A = (A_1, \dots, A_n)$, such that each agent $i \in N$ receives the bundle, $A_i \subseteq M, A_i \cap A_j = \emptyset$ for every pair of agents $i, j \in N$, and $\bigcup_{i \in [n]} A_i = M$.

Fairness and Efficiency

Definition

An allocation A is `envy-free` (EF) if it satisfies:

$$\forall i, j \in N : u_i(A_i) \geq u_i(A_j).$$

Definition

An allocation A is envy-free up to any item (EFx) if it satisfies:

$$\forall i, j \in N : \ u_i(A_i) + \min_{z \in A_j} u_i(z) \geq u_i(A_j).$$

Definition

An allocation A is pareto-optimal (PO) if there is no other allocation B such that:

$$\begin{cases} \forall \, i \in N : \ u_i(B_i) \geq u_i(A_i), \\ \exists \, j \in N : \ u_j(B_j) > u_j(A_j). \end{cases}$$

After the definitions we move to out task. We have to create the algorithm to search for EFx + PO allocations. Our concept:

1. Search for EFx allocation. If not found, return False.
2. Check the found allocation for PO.
3. If the partition is EFx + PO, return it. Otherwise, go back to the first step.

Hardness

Theorem

The problem of existence of an EFx + PO allocation is NP-hard even for two agents.

To obtain prove NP-hardness of our problem, we reduce from Partition.

Partition problem

- Input: A set of positive integers $S = s_1, \dots, s_n$.
- Question: Does there exist a partition of S into two sets S_1 and S_2 such that the sums of the numbers in the sets are equal?

Sketch of the reduction:

- Create an input for Partition with zero-weighted goods (0 for one of the agents, 1 to the other).
- Construct Partition solution from EFx and PO allocation and vice versa.

Integer Linear Programming

For simplicity of presentation, we will formulate an ILP for two agents. EFx restrictions for two agents can be simplified as follows:

$$0 \leq x_i \leq m_i, \ 0 \leq i \leq k.$$

$$\sum_{i=1}^{k} a_i x_i + \min_{j:\, m_j - x_j \neq 0} a_j \geq \sum_{i=1}^{k} a_i(m_i - x_i).$$

$$\sum_{i=1}^{k} b_i(m_i - x_i) + \min_{j:\, x_j \neq 0} b_j \geq \sum_{i=1}^{k} b_i x_i.$$

Here x_i is the variable meaning number of goods of type i that the first agent has, a_i, b_i—the utility of the object of type i for the first and the second agents, m_i—the number of goods of type i. Moreover, Pareto-optimality could be obtained in a similar fashion. It is also necessary to ensure that the search method does not return previously checked allocations. This problem can be solved by introducing additional constraints in the ILP problem.

Upper bound

Number of EFx allocations plays crucial role in the runtime estimation, so we prove lower bound and construct an example:

Lemma

The total number of allocations in the problem with m objects and two agents does not exceed $(\lceil \frac{m+k}{k} \rceil)^k$, where k is the number of different types of items.

Example

The number of EFx-allocations can be equal to $\frac{m^k}{2^k k^k}$ on some inputs with two agents.
Consider a problem with two agents and k types of items. Let the first type have value a_1 for the second player and 0 for the first player, and for all other types, the i-th item has value a_i for the first player and 0 for the second player. Let m_i be the number of items of type i and x_i be the number of items of type i held by the first player. We write the EFx condition for both players as follows:

$$a_2 x_2 + ... + a_k x_k + \min_{j: x_j \neq m_j} a_j \geq (m_2 - x_2) a_2 + ... + (m_k - x_k) a_k$$

$$(m_1 - x_1) a_1 \geq x_1 a_1$$

In the second condition, there is no minimum because it is equal to zero. It is easy to see that both conditions are satisfied when $x_1 \leq \frac{m_1}{2}$ and $x_i \geq \frac{m_i}{2}$ if $i \geq 2$. In this case, the number of EFx-allocations satisfying the condition at least $\frac{m^k}{2^k k^k}$.

Evaluation

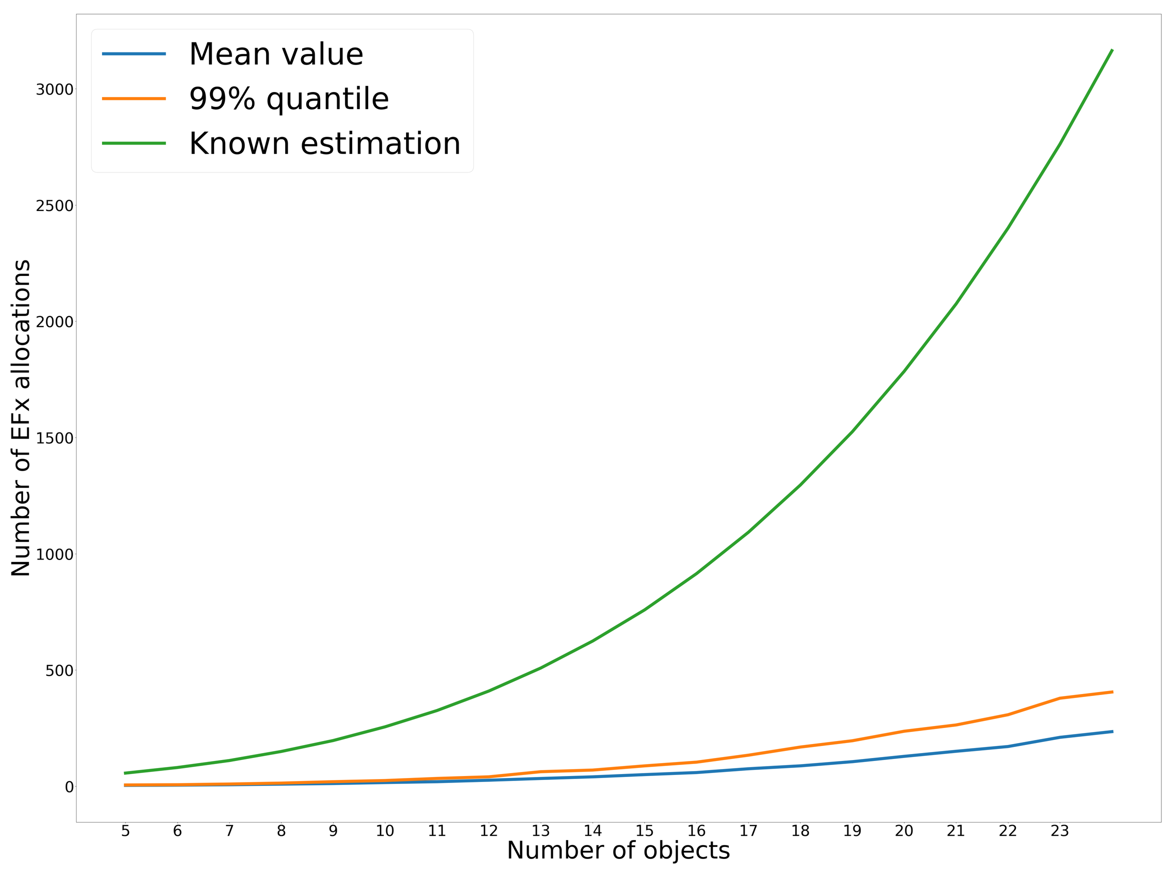

Figure 1: Number of EFx allocations for $max\ weight = 10$

At a glance, there's a significant difference between the average number of allocations and even the 99% quantile from the upper estimate, and the relative deviation only increases with the increasing number of types.
To study the impact of the weight constraint on the number of allocations, we introduced a new metric called relative difference. This is the ratio of the difference in the average number of solutions to the overall average number of solutions:

$$\text{relative difference} = \frac{\overline{solutions_1} - \overline{solutions_2}}{\overline{solutions}},$$

The main outcome is that the impact of weight differences decreases as the number of types grows. This is due to the fact that, with a small number of types, the weight difference plays a significant role that cannot be offset by the size of the sets.

Example

For instance, there are only 2 EFx allocations for an input of 2 types and 12 items:
$\{(1, 1) : 11, (1000, 1000) : 1\}$ (allocations and bundles are described in the format "type — number of items of this type"). If an agent's bundle is: $\{(1, 1) : a, (1000, 1000) : b\}$, where $a, b > 0$, they would be envied because the minimum condition from the definition of EFx will not be met.
However, by changing the boundary from 1000 to 10, a similar input becomes
$\{(1, 1) : 11, (10, 10) : 1\}$ and the number of allocations of interest to us increases.

Open questions

- Is it possible to prove some probabilistic upper bound on the number of EFx allocations?
- Is there a lower bound on the percentage of EFx allocations that are PO?

If we obtain a probabilistic estimate for the number of EFx allocations, we can then compute the expected runtime of our algorithm. We observe that this number is significantly smaller than the lower bound, suggesting the feasibility of such an estimate. The second question is potentially more significant, as it allows us to estimate the expected number of iterations. If, under certain conditions, almost every EFx allocation is also a PO allocation, then it implies that it will take only a few iterations to find an allocation.

References

[1] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, and Junxing Wang.
The unreasonable fairness of maximum nash welfare.
In *Proceedings of the 2016 ACM Conference on Economics and Computation*, EC '16, page 305–322, New York, NY, USA, 2016. Association for Computing Machinery.

[2] Georgios Amanatidis, Haris Aziz, Georgios Birmpas, Aris Filos-Ratsikas, Bo Li, Hervé Moulin, Alexandros A. Voudouris, and Xiaowei Wu.
Fair division of indivisible goods: Recent progress and open questions.
Artificial Intelligence, 322:103965, 2023.

[3] Benjamin Plaut and Tim Roughgarden.
Almost envy-freeness with general valuations.
SIAM J. Discret. Math., 34(2):1039–1068, 2020.