On the existence of EFX-allocations for more than three agents

Quentin Vermande, under supervision of Kurt Mehlhorn
Max Planck Institut für Informatik, Saarbrücken, Germany

March-July 2020

Abstract

This first year Master degree's thesis focuses on the fair allocation of indivisible goods problem, and mainly on results concerning the existence of EFX allocations. Although this is mostly a bibliographic work, some of the result presented here are new. Most of the relevant fairness and efficiency criteria in fair allocation are introduced and linked to each other. The main techniques for solving special cases of the EFX problem are given together with reasons why they do not generalize. The problem of proving the existence of EFX allocations is reduced to smaller subproblems using various proof techniques.

Contents

1 Introduction

My first year Master degree's internship took place at the Max-Planck Institute for Informatics, under the supervision of Pr. Kurt Mehlhorn. It happened shortly after the publication of an algorithm for solving the EFX allocation problem with three agents [CGM20] by members of the MPI, hence the title. Despite significant work in the field, we only have proofs of existence in reduced cases, like small number of agents or small number of goods. The main goal of this internship was thus to decide if EFX allocations always exist, i.e. finding a proof of existence or exhibiting an instance without EFX allocation.

Historically, EFX is the second relaxation of the EF (envy-free) criterion, after the EF1 criterion introduced by E. Budish in 2011. EF1 allocation always exist, but they may be unsatisfactory, as they do not prevent very large envy to appear. The EFX criterion was introduced in 2016 by Cariagannis et al. in [CKM+16]. It is stronger than EF1 and therefore rules out some of the issues caused by EF1. Unfortunately, its existence remains open.

My work concerning the EFX existence problem has been at the same time bibliographic, theoretic and practical. Bibliographic, first, as it obviously started by understanding the work that had already been done in the field before building upon it. Theoretic, because it consisted then in trying to improve the existent results and finding new ideas to explore. Practical, last, because I wrote programs in order to explore instances and allocations efficiently, find examples and counter-examples and help make conjectures.

Besides, I also worked on other project. At the beginning of my internship, I also spent some time on the physarum flow [KKM19]. Furthermore, I have been part of an immersion lab about fair division, together with Pr. Mehlhorn, Bhaskar Ray Chaudhury, Hannaneh Akrami and Golnoosh Shahkarami, in which we read papers on a wide range of subjects and worked on a few problems. This will not be further detailed as there is already much to say about EFX.

2 Allocating indivisible goods

This section defines the model for the problem of fair division of indivisible goods and gives some known results concerning the fairness and efficiency criteria.

2.1 The setting

Consider a finite set A of agents and a finite set G of goods (with $2 \leq |A|$). Each agent has a utility for any set of goods, given by the valuation function $v : A \to \mathcal{P}(G) \to \mathbb{R}_{\geq 0}$. v is called:

- general (or monotone) when: $\forall i \in A, \forall S \subset T \subset G, v_i(S) \leq v_i(T)$

- additive when: $\forall i \in A, \forall S \subset G, v_i(S) = \sum_{g \in S} v_i(\{g\})$
- non-degenerate when: $\forall i \in A, \forall S \neq T \subset G, v_i(S) \neq v_i(T)$ (or equivalently, $\forall i \in A, v_i$ is injective)
- identical when: $\forall i, j \in A, v_i = v_j$
- normalised when: $\forall i \in A, v_i(G) = 1$

Valuations will always be considered general.

An instance of the fair allocation of indivisible goods problem is (A, G, v).

A partial allocation $\mathfrak{a}$ gives, for each allocated good, the agent it is allocated to: $\mathfrak{a} : dom(\mathfrak{a}) \to A$, with $dom(\mathfrak{a}) \subset G$. $\mathfrak{a}$ is complete when $dom(\mathfrak{a}) = G$. Equivalently, we could give each agent's bundle: $B_a = (a^{-1}(\{i\}))_{i \in A} : A \to \mathcal{P}(G)$. Note that bundles are disjoint and that $\bigsqcup_{i \in A} B_{\mathfrak{a}}(i) = dom(\mathfrak{a})$ is the set of allocated goods. The valuation of $\mathfrak{a}$ is the function that gives each agent's valuation for his bundle: $v(\mathfrak{a}) = (v_i(a^{-1}(\{i\})))_{i \in A}$.

2.2 Fairness and efficiency criteria

A fairness or efficiency criteria is a property of allocations. In order to define approximate criteria, let $\alpha \in \mathbb{R}_{\geq 0}$. Let $\mathfrak{a} : G \to A$.

- $\mathfrak{a}$ is α-envy-free (α-EF) when no agent envies another one up to a factor of α:

$$\forall i, j \in A, \alpha v_i(B_{\mathfrak{a}}(j)) \leq v_i(B_{\mathfrak{a}}(i))$$

- $\mathfrak{a}$ is α-envy-free up to one good (α-EF1) when no agent envies another one (up to a factor of α), possibly after removing some chosen good:

$$\forall i, j \in A, B_{\mathfrak{a}}(j) = \varnothing \vee \exists g \in B_{\mathfrak{a}}(j), \alpha v_i(B_{\mathfrak{a}}(j) \setminus \{g\}) \leq v_i(B_{\mathfrak{a}}(i))$$

- $\mathfrak{a}$ is α-envy-free up to any good (α-EFX) when no agent envies another one (up to a factor of α), possibly after removing any good (in the envied agent bundle):

$$\forall i, j \in A, \forall g \in B_{\mathfrak{a}}(j), \alpha v_i(B_{\mathfrak{a}}(j) \setminus \{g\}) \leq v_i(B_{\mathfrak{a}}(i))$$

- $\mathfrak{a}$ is α-proportional (α-Prop) when each agents reveives at least a fraction $\frac{\alpha}{|A|}$ of the utility he has for G:

$$\forall i \in A, \frac{\alpha}{|A|} v_i(G) \leq v_i(B_{\mathfrak{a}}(i))$$

- The maximin share of an agent $i \in A$ is the maximum utility he can obtain by cutting G into $|A|$ (possibly empty) parts and taking the least valuable one:

$$\text{MMS}_i = \max_{a : G \to A} \min_{j \in A} v_i(B_{\mathfrak{a}}(j))$$

  $\mathfrak{a}$ is α-maximin (α-MMS) if each agent receives at least α times his maximin share:

$$\forall i \in A, \alpha \text{MMS}_i \leq v_i(B_{\mathfrak{a}}(i))$$

- The Nash social welfare (NSW) of $\mathfrak{a}$ is:

$$\text{NSW}(\mathfrak{a}) = ||v(\mathfrak{a})||_0 = (\prod_{i \in A} v_i(B_{\mathfrak{a}}(i))^{\frac{1}{|A|}}$$

  $\mathfrak{a}$ is α-maximum Nash social welfare (α-MNW) when NSW($\mathfrak{a}$) is at least α times the maximum Nash social welfare:

$$\alpha \max_{b:G \to A} \text{NSW}(b) \leq \text{NSW}(\mathfrak{a})$$

- An allocation $\mathfrak{a} : G \to A$ is said to Pareto-dominate an allocation b when $v(b) \leq v(\mathfrak{a})$ (with $\leq$ the pointwise inequality) and $v(b) \neq v(\mathfrak{a})$. $\mathfrak{a}$ is Pareto-optimal (PO) when no allocation Pareto-dominates $\mathfrak{a}$:

$$\forall b : G \to A, v(\mathfrak{a}) \leq v(b) \Rightarrow v(\mathfrak{a}) = v(b)$$

- For $\mathfrak{a} : G \to A$, let $\text{sort}(\mathfrak{a}) \in \mathbb{R}_{\geq 0}^{|A|}$ be the sorted vector containing all values in $v(\mathfrak{a})$, i.e. sort($\mathfrak{a}$) is the increasing vector such that $\forall x \in \mathbb{R}_{\geq 0}, |v(\mathfrak{a})^{-1}(\{x\})| = |\text{sort}(\mathfrak{a})^{-1}(\{x\})|$. $\mathfrak{a}$ is leximin when sort($\mathfrak{a}$) is maximum among all allocation with respect to the lexicographic order $\leq_{\text{lex}}$ on $\mathbb{R}_{\geq 0}^{|A|}$:

$$\forall b : G \to A, \text{sort}(b) \leq_{\text{lex}} \text{sort}(\mathfrak{a})$$

2.3 Known relations between the criteria

EF is arguably the strongest criterion, as we have tho following implication chains: α-EF$\Rightarrow$ α-EFX$\Rightarrow$ α-EF1 and $\frac{|A|-1}{\frac{|A|}{\alpha}-1}$-EF $\Rightarrow$ α-Prop$\Rightarrow$ α-MMS.

Proof.

All implications are straightforward, except $\frac{|A|-1}{\frac{|A|}{\alpha}-1} - EF \Rightarrow \alpha$-Prop. Let $\mathfrak{a}$ be a non-α-Prop allocation. Let $i \in A$ such that $v_i(B_{\mathfrak{a}}(i)) < \frac{\alpha}{|A|}$. By the pigeonhole principle, there exists $j \in A$ such that $v_i(B_{\mathfrak{a}}(j)) \geq \frac{1}{|A|-1}(1 - \frac{\alpha}{|A|})$. We have $\frac{v_i(B_{\mathfrak{a}}(i))}{v_i(B_{\mathfrak{a}}(j))} < \frac{\frac{\alpha}{|A|}}{\frac{1}{|A|-1}(1-\frac{\alpha}{|A|})} = \frac{|A|-1}{\frac{|A|}{\alpha}-1}$. Thus $\mathfrak{a}$ is not $\frac{|A|-1}{\frac{|A|}{\alpha}-1}$-EF.

It has been shown in [CKM+16] that the following holds: MNW $\Rightarrow$ EF1 $\wedge$ PO. We also know from [KPW18] that: EFX $\Rightarrow$ $\frac{4}{7}$MMS.

3 EFX allocation

3.1 Envy-graph

The most useful tool when searching for EF, EFX or EF1 is the envy graph. Let $\mathfrak{a}$ be a partial allocation $\mathfrak{a} : \text{dom}(\mathfrak{a}) \to A$. The envy graph of $\mathfrak{a}$ is the following:

$$E_a = (A, \{(i,j) \in A^2, v_i(B_{\mathfrak{a}}(i)) < v_i(B_{\mathfrak{a}}(j))\})$$

In words, there is an edge from agent i to agent j in E_a if and only if i envies j in allocation $\mathfrak{a}$. Envy graphs are very useful thanks to the following lemma [CKMS19], [CGM20], [KPW18], [ANM19]:

Lemma 1 (decyclifying the envy graph)**.** *Let $\mathfrak{a}$ be an EF (resp. EFX, EF1) partial allocation. Then there exists an EF (resp. EFX, EF1) partial allocation $b : dom(\mathfrak{a}) \to G$ such that E_b is acyclic.*

Proof.

The proof is algorithmic. Consider $b = a$. While E_b has a cycle C, shift the bundles along C, with C as small as possible. The valuation of each agent for his own bundle increases, and even strictly for agents in C. Thus the new allocation Pareto-dominates b. As there is a finite number of allocations with domain $dom(\mathfrak{a})$, the algorithm must stop.

We can give more information on the envy graph by giving weight to the edges. To this end, we quantify the (strong) envy of an agent i towards an agent j by:

$$envy_{\mathfrak{a}}(i,j) = \min(\{k \in \mathbb{N}, \forall S \subset B_{\mathfrak{a}}(j), |S| = k \Rightarrow v_i(B_{\mathfrak{a}}(j) \setminus S) \leq v(\mathfrak{a})_i\})$$

In words, $envy_{\mathfrak{a}}(i,j)$ is the minimum number of goods one must remove from $j's$ bundle so that i does not envy j anymore, removing the least valuable goods according to i. $envy_a$ is well defined as, for $i, j \in A$, $|B_{\mathfrak{a}}(j)|$ is in the above defined set. Note that $\mathfrak{a}$ is EF when $envy_a = 0$ and EFX when $im(envy_{\mathfrak{a}}) \subset \{0, 1\}$.

3.2 Injective and normalised valuations

To begin with, we note that the EF, EFX and EF1 criteria only depend on the orders induced by the valuations over $\mathcal{P}(G)$:

Lemma 2. *Let $v, v' : A \to \mathcal{P}(G) \to \mathbb{R}_{\geq 0}$ inducing the same orders over $\mathcal{P}(G)$ ($\forall i \in A, \forall S, T \subset G, v_i(S) \leq v_i(T) \Rightarrow v'_i(S) \leq v'_i(T)$) and $\mathfrak{a}$ a partial allocation. If $\mathfrak{a}$ is EF (resp. EFX, EF1) in (A, G, v), then $\mathfrak{a}$ is EF (resp. EFX, EF1) in (A, G, v').*

Corollary. *If any non-degenerate and normalised instance has an EFX allocation, then any instance has an EFX allocation.*

Proof.

Let (A, G, v) be any instance. We construct a non-degenerate valuation v' algorithmically. Start with $v' = v$. While there exist $S \neq T \subset G$ such that $v'(S) = v'(T)$, let $\delta = \min_{\substack{X,Y \subset G \\ v'(X) \neq v'(Y)}} |v'(X) - v'(Y)|$ and set $v'(S) \leftarrow v'(S) + \frac{1}{2}\delta$. The cardinality of the equivalence classes of $\{(X,Y) \in \mathcal{P}(G)^2, v'(X) = v'(T)\}$ decrease, by one for one class. Thus the algorithm terminates. Furthermore, this algorithm maintains the following invariant: $\forall S, T \subset G, v'(S) < v'(T) \Rightarrow v(S) \leq v(T)$.

We have an EF (resp. EFX, EF1) allocation $\mathfrak{a}$ for (A, G, v'). With lemma 2, $\mathfrak{a}$ is EF (resp. EFX, EF1) for (A, G, v).

4 Algorithms

4.1 EFX for three agents

In this subsection, we only consider additive and non-degenerate valuations. There exists an algorithm, described in [CGM20], that finds an EFX allocation. This algorithm increases the valuation of the current allocation with a variant of the Pareto potential, as long as there is an unallocated good. It uses two tools: the envy graph of the current allocation and the champions graph [CKMS19], [CGM20]. Consider a partial allocation $\mathfrak{a}$ and an unallocated good. The champions graph for $\mathfrak{a}$ and g is:

$$M_{a,g} = (A, \{(i,j) \in A^2, i \in \underset{k \in A}{\operatorname{argmin}} \inf_{\substack{S \subset B_{\mathfrak{a}}(j) \cup \{g\} \\ v(\mathfrak{a})_k < v_k(S)}} |S|\})$$

In words, agent i champions agent j when he needs the least number of goods in $B_{\mathfrak{a}}(j) \cup \{g\}$ to increase his valuation, among all agents. It is not hard to see that every agent is championned, and thus that the champions graph is cyclic. One of the most important results for the correctness of the algorithm generalises to any number of agents [CGM20]:

Lemma 3. *Let $\mathfrak{a}$ be a partial EFX allocation and g an unallocated good. Let $i, j \in A$. If $(i,j) \in M_{a,g}$ and there exists a path from j to i in E_a, then there exists an EFX allocation Pareto dominating $\mathfrak{a}$ (in which j's valuation strictly increases).*

This lemma reduces considerably the number of pairs envy graph-champions graph that remain to consider. This could let us hope of a generalisation of the above mentioned algorithm. Unfortunately, there is a setting with 4 agents and 11 goods, a partial EF allocation and an unallocated good such that there exists only one cycle in the champions graph and any Pareto dominating allocation breaks bundles of agents outside this cycle:

	1	2	3	4	5	6	7	8	9	10	11	$B_{\mathfrak{a}}(1)$	$B_{\mathfrak{a}}(2)$	$B_{\mathfrak{a}}(3)$	$B_{\mathfrak{a}}(4)$
v_1	10	20	20	11	30	20	0	50	0	50	0	51	50	50	50
v_2	10	0	20	30	20	20	11	51	0	51	0	50	51	51	51
v_3	0	27	0	0	25	26	0	30	21	51	0	27	51	51	51
v_4	0	0	2	2	0	0	50	51	0	21	30	4	50	51	51
$\mathfrak{a}$		0	0	0	1	1	1	2	2	3	3				

Here, there is only one unallocated good (1). The four rightmost columns of the table show that $\mathfrak{a}$ is indeed EF. The only cycle in the champions graph is the cycle $(1,2)$. But any EFX allocation Pareto-dominating $\mathfrak{a}$ changes $\mathfrak{a}_{|B_{\mathfrak{a}}(3) \cup B_{\mathfrak{a}}(4)}$.

4.2 Partitions

Instead of searching directly for an EFX allocation, we can search for a partition of G into $|A|$ bundles that one can allocate to the agents in order to obtain an EFX allocation. As some bundles may be empty, we need to take another definition of partition than the standard one: a partition of G along A is a function $P : A \to \mathcal{P}(G)$ such that $G = \bigsqcup_{i \in A} P_i$. The set of partitions of G along A will be denoted $\prod_A G$. Note that, for $\mathfrak{a} : G \to A$, $B_a \in \prod_A G$. There is a simple criterion to decide whether the allocation of the bundles of a partition P is EFX: all agents must receive a minimum admissible value, defined as follows:

$$mAV_P(i) = \max_{p \in dom(P)} \sup_{g \in P_p} v_i(P_p \setminus \{g\})$$

Lemma 4. *An allocation $\mathfrak{a} : G \to A$ is EFX if and only if $\forall i \in A, mAV_{B_a}(i) \leq v(\mathfrak{a})_i$.*

As a consequence, we have the following:

Corollary. *A partition* $P \in \prod_A G$ *induces an EFX allocation if and only if his allocation graph* $(A \sqcup A, \{(i,p) \in A^2, mAV_P(i) \leq v_i(P_p)\})$ *has a perfect matching.*

It is easy to find a partition such that no node in its allocation graph is isolated: any partition maximizing $(||(v_i(P_p))_{i\in A}||_\infty)_{p\in A}$ (lexicographcally ordered) or $||(||(v_i(P_p))_{i\in A}||_\infty)_{p\in A}||_0$ suits. These are the natural generalizations of leximin and MNW allocation.

Unfortunately, all considered criteria are sensible to an agent having higher valuations than the others. Because of that, they fail in the following setting (with $4 \leq n = |A|$ and $\varepsilon \in \mathbb{R}^{+*}$ small enough):

	Good type 1	Good type 2	Good type 3	Good type 4	agents
Valuation type 1	0	$\frac{1}{3(n-1)}$	$\frac{1}{3(n-1)}$	$\frac{1}{6(n-1)}$	$\{1\}$
Valuation type 2	$1-2(2(n-1)-1)\varepsilon$	$(n-1)\varepsilon$	ε	ε	$[\![2,n]\!]$
Goods	$\{1\}$	$\{2\}$	$[\![3,n]\!]$	$[\![n+1,2n-1]\!]$	

In any partition with an EFX allocation, goods of type 1 or 2 are alone because of the agents of type 2. In the partitions maximizing our criterias, the good of type 2 is allocated together with a good of type 4 because the partition needs to be close to the MMS partition of the agent of type 1 without the good of type 1.

4.3 Decreasing paths

Another way to look at the problem is to start from a complete allocation and try to modify it until reaching an EFX allocation. We define a potential to quantify the strong envy in an allocation, such as $\sum_{i\neq j\in A}(envy_{\mathfrak{a}}(i,j)-1)_+$. This potential should be optimal only on EFX allocation. For instance, the above defined one is non negative and has value 0 if and only if the allocation is EFX. Then we consider udate rules, which is equivalent to a set of edges in the graph of all allocations. The most general graph considered is the one containing an edge from $\mathfrak{a} : G \to A$ to $b : G \to A$ when b is obtained from $\mathfrak{a}$ by decyclifying the envy graph or if there exists a simple path in (A, A^2) such that b is reached when all agents in the path give some good to the next one.

Unfortunately, this graph may contain local optima that are not EFX. Consider the following instance (with $A = [\![1,3]\!]$ and $G = [\![1,5]\!]$):

	1	2	3	4	5	$B_{\mathfrak{a}}(1)$	$B_{\mathfrak{a}}(2)$	$B_{\mathfrak{a}}(3)$
v_1	4	32	26	12	4	44	8	46
v_2	8	19	42	1	9	20	17	42
v_3	15	11	26	19	8	30	23	46
$\mathfrak{a}$	1	0	2	0	1			

4.4 Two-value instances

Although MNW does not implies EFX in the general case, the implication is true when the valuations can take only two values, as stated by theorems 3.1 and 3.2 in [ABF+20]. Computing a MNW allocation as known to be polynomial for two-value instances when one of the values is 0 and to be NP-hard. The case of two-values instances is still open. Computing an EFX allocation in a two-value instance can still be done in polynomial time, with the following algorithm (from [ABF+20]):

Algorithm 1 Match&Freeze

Input: a two-value instance $(A = [\![1, n]\!], G, v)$ with values $0 \leq a < b$.

```
𝔞 ← ∅
L ← A
U ← G
l = (1, ..., n)
while U ≠ ∅ do
    Let H = (L ⊔ U, {(i, g) ∈ L × U, v_i(g) = b}).
    Let M a maximum matching on H.
    for (i, g) ∈ M do
        𝔞 ← a ∪ {(g, i)}.
        U ← U \ {g}.
    for i ∈ l do
        if i ∉ M then
            Let g ∈ U.
            𝔞 ← a ∪ {(g, i)}.
            U ← U \ {g}.
    Let F = {i ∈ L, i is reachable from H \ M in H}.
    Remove agents in F from L for the next ⌊b/a⌋ − 1 rounds.
    Put agents in F at the end of l.
return 𝔞.
```

This kind of greedy algorithms can not generalize, because of instances like the following: Let $M, x, \varepsilon, M', x', \delta \in \mathbb{R}^{+*}$ and the instance (with $2 \leq n = |A| < m = |G|$) :

	Good type 1	Good type 2	Good type 3	Good type 4	Agents
Valuation type 1	M	x	ε	ε	$\{1\}$
Valuation type 2	x'	δ	M'	δ	$[\![2, n]\!]$
Goods	$\{1\}$	$\{2\}$	$[\![3, n]\!]$	$[\![n+1, m]\!]$	

1. If $\varepsilon < x$, $x + (m - n - 1)\varepsilon < M$, $x' < (m - n)\delta$ and $x' < M'$, then, in any complete EFX allocation, agent 1 gets good 1.

2. If $\varepsilon < x < M$, $\delta < x' < M'$ and $M + \delta < x + x'$, then $\max\limits_{a \in 2^{A \times G}} \sum\limits_{i \in A} \max\limits_{B_{\mathfrak{a}}(i)} v_i = x + x' + (n-2)M'$.

Proof.

1. *Suppose by contradiction that we have a complete allocation $\mathfrak{a}$ such that $1 \notin B_{\mathfrak{a}}(1)$. We can assume w.l.o.g. that $1 \in B_{\mathfrak{a}}(2)$.*
    - *Suppose $B_{\mathfrak{a}}(2) = \{1\}$. Since $\mathfrak{a}$ is EFX and $x' < M'$, each agent which recieves a good of type 3 does not recieve anything else. Thus one agent, say i, gets all goods of type 2 and 4. But $v_2(B_{\mathfrak{a}}(2)) = x' < (m - n + 1)\delta = v_1(B_{\mathfrak{a}}(i) \setminus \{1\})$, which contradicts the fact that $\mathfrak{a}$ is EFX.*
    - *Suppose $\{1\} \subsetneq B_{\mathfrak{a}}(2)$. Agent 1 has two goods and no good has value 0 for anyone, so every agent recieves at least one good. Thus $v_1(B_{\mathfrak{a}}(1)) \leq x + (m-n-1)\varepsilon < M = v_1(\{1\})$ and $\mathfrak{a}$ is not EFX*

In each case, $\mathfrak{a}$ is not EFX.

2. *To maximise the sum, $n-2$ agents with valuation type 2 recieve a good of type 3 and the two remaining agents share the goods of type 1 and 2. As $M+\delta < x + x'$, agent 1 receives good 2 and the last agent good 1.*

From this, we can deduce that any algorithm that starts with a maximum matching and does never deallocate goods will not always return an EFX allocation. The same holds for an algorithm which starts with each agent taking in turn his most preferred good, because in the previous instance, if agent 1 chooses last, good 1 is already allocated.

5 Known cases

5.1 Identical valuations

The following result is a variant of one found in [PR17a], where a variant of leximin allocations are introduce in order to deal with degenerate valuations:

Theorem 1. *When valuations are non-degenerate and identical, any leximin allocation is EFX.*

Proof.

Let $\mathfrak{a}$ be a non-EFX allocation. Let $i, j \in A$ such that $2 \leq envy_{\mathfrak{a}}(i,j)$. We have $g \in B_{\mathfrak{a}}(j)$ such that $v(\mathfrak{a})_i < v_i(B_{\mathfrak{a}}(j)) \setminus \{g\})$ (as otherwise we would have $envy_{\mathfrak{a}}(i,j) \leq 1$). Let's show that allocating g to i yelds a better instance with respect to the leximin criterion. Let $b = a_{|G\setminus\{g\}} \cup \{(g,i)\}$. Let $A' = \{k \in A, v(\mathfrak{a})_k \leq v(\mathfrak{a})_i\} \setminus \{i\}$. We have $\forall k \in A \setminus \{i,j\}, v(\mathfrak{a})_k = v(b)_k$, $v(\mathfrak{a})_i < v(b)_i$ and $v(\mathfrak{a})_j < v(b)_j$. Thus we can derive $sort(\mathfrak{a})_{||A'|} = sort(b)_{||A'|}$ and $sort(\mathfrak{a})_{|A'|} < sort(b)_{|A'|}$. We can conclude $sort(\mathfrak{a}) <_{lex} sort(b)$ and $\mathfrak{a}$ is not leximin.

5.2 Reductions

In this section we only consider non-degenerate valuations. In a few cases, it is easy to find an EFX allocation or to reduce to a smaller instance of the problem:

Lemma 5. *If $|G| \leq |A| + 1$, then there exists an EFX allocation.*

Proof.

Each agent picks in turn his most preferred good. The last good (if it exists) is given to the last agent who received a good.

Proposition 1. *If valuations are additive and all agents have the same ranking over goods, then there exists an EFX allocation.*

Proof.

Alternate between letting each source in the envy graph choose his most preferred remaining good and eliminating cycles in the envy-graph.

Note that the above described algorithm always gives an EF1 allocation (even with general valuations). The algorithm given in the last two proofs (which are in fact the same) do not give better results than those stated because of the following: assume we have given agent i his most preferred good. When agent j receives a good, i can not strongly envy j, as $|B_{\mathfrak{a}}(j)| = 1$. When j receives a second good, i did not envy j, and the good j receives had lower value to i than $i's$ good. Thus i still does not envy j. The problem occurs when j receives a third good, if this good has higher value to i than $j's$ least valuable good, because in does not change $\min_{B_{\mathfrak{a}}(j)} v_i$ but $v_i(B_{\mathfrak{a}}(j))$ increases. In the case when $|G| = |A| + 2$, a workaround was found for additive valuations:

Theorem 2 ([ANM19])**.** *When the valuations are additive and $|G| \leq |A| + 2$, there exists an EFX allocation.*

Proof.

The case $|G| \leq |A| + 1$ has already been adressed. In the case $|G| = |A| + 2$, we give $|A| - 1$ agents in turn their most preferred good. We are left with three goods g_1, g_2, g_3. Introduce two virtual goods p, q with:

- $\forall i \in A, v_i(q) = \min(\{v_i(g_1), v_i(g_2), v_i(g_3)\})$
- $\forall i \in A, v_i(p) = v_i(\{g_1, g_2, g_3\}) - v_i(q)$

Give p to the agent without good, decyclify the envy graph and give q to a source. The owner of p receives his two most preferred good in $\{g_1, g_2, g_3\}$ and the owner of q the remaining one. Two cases occur:

- *If two agents i and j have two goods, one of them had nothing when creating p and q, say i. As every agent except i got his most preferred good at his turn, it is easy to see that only i can strongly envy j. When p is allocated, there is no strong envy (for the same reason and as it is allocated to i). After decyclifying, there is still no strong envy. The agent who receives q has two goods, but he was not envied and everyone thinks he has a better good than q, so there is still no strong envy. If i received p, valuations according to him do not change when allocating the real goods. If i received q and another good, his valuation can only increase and $j's$ valuation decrease, so there is still no strong envy.*
- *If an agent i received three goods, he must have received p and q. When allocating q, everyone thinks q is the least valuable good in $i's$ bundle and i was not envied, thus i is not strongly envied. Allocating the real goods does not change the valuations, thus i is still not strongly envied. As all other agents have only one good, it is easy to see that no one is strongly envied.*

In both cases, the resulting allocation is EFX.

The original result in [ANM19] is in fact stronger (with the same algorithm), but it uses a fairness criterion that will not be useful anywhere else in this report and the proof for EFX only is far simpler than the one in the paper.

Lemma 6. *If valuations are additive, there exists $i \in A$ such that $v_i(G) - (n-1)\min_G v_i - \max_G v_i \leq \max_G v_i$ and $(A' = A \setminus \{i\}, G' = G \setminus \underset{G}{argmax}\ v_i, (v_{j|G'})_{j \in A'})$ has an EFX allocation, then (A, G, v) has an EFX allocation.*

Proof.

With lemma 5, we can assume that $|A|+2<|G|$. Let $g \in \underset{G}{argmax}\ v_i$ and $\mathfrak{a}$ an EFX-allocation for the reduced instance. Let $b = a \cup \{(g,i)\}$. As i has only one good, no one strongly envies him. Thus it only remains to show that i does not envy anyone to conlude that b is EFX. Let $j \in A \setminus \{i\}$ and $h \in B_\mathfrak{a}(j)$ (if $B_\mathfrak{a}(j) = \varnothing$, i does obviously not envy j). As v is non-degenerate, every agent has at least one good in $\mathfrak{a}$. Thus:

$$\begin{aligned} v_i(B_\mathfrak{a}(j) \setminus \{h\}) &= v_i(G \setminus \bigcup_{\substack{k\in A\\ k\neq j}} B_\mathfrak{a}(k) \setminus \{h\}) \\ &= v_i(G) - \sum_{\substack{k\in A\\ k\neq j}} v_i(B_\mathfrak{a}(k)) - v_i(h) \\ &\leq v_i(G) - (n-1)\min_G v_i - \max_G v_i \\ &\leq \max_G v_i \\ &= v(\mathfrak{a})_i \end{aligned}$$

Thus i does not strongly envy j.

We can conclude that b is EFX.

One way to reduce any instance is to merge two goods, summing up their utility. If the new good is allocated to an envied agent in an EFX allocation, the envy might become strong after unmerging, and there is no way we have found to change the allocation in order to obtain an EFX allocation. On the other hand, if the new good was allocated to an unenvied agent, this agent is still unenvied after unmerging the goods. Thus we would like to know which goods can be allocated to an unenvied agent. Unfortunately, we have not found any property of such goods. For example, take $|G| = 2|A| - 1$, identical additive valuations and all goods have value 2 except one, say g, that has value 3. The only EFX allocations give one agent good g and all other agents two goods. In these allocations, the only unenvied agent is the one that has good g. Thus g, which is the most valuable good to every agent, is the only good that can be allocated to an unenvied agent in an EFX allocation.

6 The existence problem

6.1 Number of instances

As stated in lemma 2, two instances (A, G, v) and (A, G, v') can be considered equivalent when, for $i \in A$, v_i and v'_i induce the same order over $\mathcal{P}(G)$. Thus we only need the number $\mathfrak{a}_{|G|}$ of linear extension of the subset order on $\mathcal{P}(G)$ to count the number of (morally different) instances given A and G. Even though no expression is known for this number, bounds are given in [SK87]:

Theorem 3. *For $n \in \mathbb{N}$, we have:* $\prod_{i=0}^{n} \binom{n}{i}! \leq a_n \leq \exp(2^n \frac{6\ln n}{n}) \prod_{i=0}^{n} \binom{n}{i}!$

From this, one can deduce the following asymptotic expansion:

Theorem 4. $\mathfrak{a}_n \underset{n\to+\infty}{=} \ln(\binom{n}{\lfloor \frac{n}{2} \rfloor}) - \frac{3}{2} + O(\frac{\ln n}{n})$

In order to prove this theorem, one wants to use convexity inequalities. Unfortunately, applying these inequalities directly is not tight enough:

Lemma 7. *1.* $\sum_{k=1}^{n} \binom{n}{k} \frac{1}{k} = 2^{n+1}(\frac{1}{n} + \frac{1}{n^2} + O(\frac{1}{n^3}))$

2. $\sum_{k=1}^{n-1} \binom{n}{k} \frac{1}{k(n-k)} = \frac{2}{n} \sum_{k=1}^{n-1} \binom{n}{k} \frac{1}{k}$

3. $\sum_{k=1}^{n} \binom{n}{k} \frac{1}{k^2} = O(\frac{2^n}{n^2})$

4. $\frac{1}{2^n} \sum_{k=1}^{n} \binom{n}{k} \ln(k) = 2\ln n - 2\ln 2 - \frac{1}{n} + O(\frac{1}{n^2})$

5. $\ln(n!) = n\ln n - n + \frac{1}{2}\ln n + \frac{1}{2}\ln(2\pi) + O(\frac{1}{n})$

6. $\frac{1}{2^n} \sum_{k=0}^{n} \ln(\binom{n}{k}!) = n\ln 2 - \frac{1}{2}\ln n - \frac{3}{2} + \frac{1}{2}\ln(\frac{2}{\pi}) + O(\frac{1}{n})$

7. $\ln(\binom{n}{\lfloor \frac{n}{2} \rfloor}) = n\ln 2 - \frac{1}{2}\ln n + \frac{1}{2}\ln(\frac{2}{\pi}) + O(\frac{1}{n})$

Proof.

Except for point 4. all expansions can be computed directly, using only natural tricks (and the preceding expansions). Now we prove point 4.

Let $u = (\frac{1}{2^n} \sum_{k=1}^{n} \binom{n}{k} \ln(k))_{n \in (\mathbb{Z}_{\geq 0})}$. *First we write* $u_n, n \in (\mathbb{Z}_{\geq 0})$ *in a more practical way. We have, for* $n \in (\mathbb{Z}_{\geq 0})$:

$$\begin{aligned} 2^{n+1} u_n &= 2\sum_{k=1}^{n} \binom{n}{k} \ln(k) \\ &= 2\ln(n) + \sum_{k=1}^{n-1} \binom{n}{k} \ln(k) + \sum_{k=1}^{n-1} \binom{n}{n-k} \ln(k) \\ &= 2\ln(n) + \sum_{k=1}^{n} \binom{n}{k} \ln(k) + \sum_{i=0}^{n-1} \binom{n}{i} \ln(n-i) \\ &= 2\ln(n) + \sum_{k=1}^{n} \binom{n}{k} \ln(k(n-k)) \end{aligned}$$

We now prove bounds on $u_n, n \in (\mathbb{Z}_{\geq 0})$ *using the concavity of* $\ln$.

For the upper bound:

$$\begin{aligned} 2u_n &= \frac{1}{2^n} \sum_{k=1}^{n} \binom{n}{k} \ln(k(n-k)) + O(\frac{\ln(n)}{2^n}) \\ &\leq \frac{2^n-2}{2^n} \ln(\frac{1}{2^n-2} \sum_{k=1}^{n} \binom{n}{k} k(n-k)) + O(\frac{\ln(n)}{2^n}) \\ &\leq \frac{2^n-2}{2^n} \ln(\frac{1}{2^n-2} \sum_{k=1}^{n} \binom{n}{k}(-k(k-1) + (n-1)k)) + O(\frac{\ln(n)}{2^n}) \\ &\leq \frac{2^n-2}{2^n} \ln(\frac{1}{2^n-2}(-\sum_{k=2}^{n} n(n-1)\binom{n-2}{k-2} + (n-1)\sum_{k=1}^{n} n\binom{n-1}{k-1})) + O(\frac{\ln(n)}{2^n}) \\ &\leq \frac{2^n-2}{2^n} \ln(\frac{1}{2^n-2}(-n(n-1)2^{n-2} + n(n-1)2^{n-1})) + O(\frac{\ln(n)}{2^n}) \\ &\leq \frac{2^n-2}{2^n} \ln(\frac{1}{2^n-2} n(n-1)2^{n-2}) + O(\frac{\ln(n)}{2^n}) \\ &= \frac{2^n-2}{2^n}(2\ln n - 2\ln 2 + \ln(1-\frac{1}{n}) - \ln(1-\frac{1}{2^{n+1}})) + O(\frac{\ln(n)}{2^n}) \\ &= 2\ln n - 2\ln n - \frac{1}{n} + O(\frac{1}{n^2}) \end{aligned}$$

Now we obtain the lower bound:

$$\begin{array}{rcl} -2u_n & = & \frac{1}{2^n}\sum_{k=1}^{n-1}\binom{n}{k}\ln(\frac{1}{k(n-k)}) + O(\frac{\ln(n)}{2^n}) \\ & \leq & \frac{2^n-2}{2^n}\ln(\frac{1}{2^n-2}\sum_{k=1}^{n-1}\binom{n}{k}\frac{1}{k(n-k)}) + O(\frac{\ln(n)}{2^n}) \\ & = & \frac{2^n-2}{2^n}\ln(\frac{1}{2^n-2}\frac{2}{n}2^{n+1}(\frac{1}{n}+\frac{1}{n^2}+O(\frac{1}{n^3})) + O(\frac{\ln(n)}{2^n}) \\ & = & \frac{2^n-2}{2^n}(-2\ln n + 2\ln 2 + \ln(1+\frac{1}{n}+O(\frac{1}{n^2})) - \ln(1-\frac{1}{2^{n+1}})) + O(\frac{\ln(n)}{2^n}) \\ & = & -2\ln n + 2\ln 2 + \frac{1}{n} + O(\frac{1}{n^2}) \end{array}$$

Thus, we conclude $u_n = \ln n - \ln 2 - \frac{1}{n} + O(\frac{1}{n^2})$.

This shows that the number of instances to consider increases very fast. In fact, even for small instances, the space of instances is huge (see sequence A046873 in OEIS).

6.2 Induction

We are mainly interested in showing that EFX allocations always exist. With this in mind, a mathematical proof of existence instead of an algorithm would be sufficient. One of the most natural way of adressing the problem mathematically is through a proof by induction, or equivalently considering a smallest instance without EFX allocation. Note that the reductions stated section 5.2 give information on this minimal instance. There are two natural ways of reducing an instance:

- One can remove a good, consider an EFX allocation in the reduced instance and try to add the removed good. This is the point of view considered in most algorithms, such as the one for three agents [CGM20].
- One can merge two goods (giving the sum of the utilities of the two goods to the new one), consider an EFX allocation in the reduced instance and unmerge the two goods. In the case when unmerging creates strong envy, only one agent is strongly envied, namely the owner of the unmerged goods. Furthermore, if this agent is unenvied in the reduced allocation, he is still unenvied in after unmerging and thus the allocation remains EFX. Thus, we have two reduced problems.

Lemma 8. *If there exists an algorithm taking an allocation* $\mathfrak{a}$ *such that at most one agent is strongly envied and* $im(envy_{\mathfrak{a}}) \subset \{0,1,2\}$ *and returning an EFX allocation, then there always exists an EFX allocation.*

Lemma 9. *If there exists an algorithm taking an instance and returning two goods such that, after merging them, the new good can be allocated to an unenvied agent in an EFX allocation, then there always exists an EFX allocation.*

Unfortunately, characterising goods that can be allocated to a source in an EFX allocation is challenging. Consider the two following insances:

- $2 \leq |A|, |G| = |A|$, valuations are additive, identical and non-degenerate. Then the only good that can be allocated to a source in an EFX allocation is the least valuable one (because in any EFX allocation, each agent receives only one good)

- $2 \leq |A|, |G| = 2|A| - 1$, valuations are additive and identical, all goods have utility 2 except one which has utility 3. Then the only good that can be allocated to a source in an EFX allocation is the most valuable one (because in any EFX allocation, all agents are given two goods exept one that has only one good and this good must be the most valuable one).

If we still do not know how to prove that an instance has an EFX allocation knowing EFX allocations of smaller instances, it is rather easy to construct bigger instances without EFX allocation having one.

Lemma 10. *There exists $f : \mathbb{Z}_{\geq 0} \to \overline{\mathbb{Z}_{\geq 0}}$ such that, for $n, m \in \mathbb{Z}_{\geq 0}$:*

- *if $m < f(n)$, then every instance with n agents, m goods and additive valutions admits an EFX allocation*
- *if $f(n) \leq m$, then there exists an instance with n agents, m goods and additive valutions that admits no EFX allocation*

Proof.

Let $n \in \mathbb{Z}_{\geq 0}$. We want to show that $S = \{m \in \mathbb{Z}_{\geq 0}$, there exists an instance with n agents and m goods that admets no EFX allocation$\}$ is of the form $[\![x, +\infty[\![$. It suffices to show that $\forall m \in S, m+1 \in S$. Let $m \in S$ and (A, G, v) an instance with $|G| = m$ and no EFX allocation. Let $g \notin G$. Then it is immediate that $(A, G \sqcup \{g\}, (v_i \cup (v_i(H \setminus \{g\}))_{\substack{H \subset G \sqcup \{g\} \\ g \in H}})_{i \in A})$ has no EFX allocation, and thus $m + 1 \in S$.

Note that we have equivalent results as the previous ones when restricting to additive instances, non-degenerate instances and additive and non-degenerate instances.

6.3 Topology

In this section, we consider the set of instances, with A and G fixed, as a topological space. Let $X = A \to \mathcal{P}(G) \to \mathbb{R}_{\geq 0}$. We equip X with the topology defined by $|| \cdot ||_\infty$. Let $E = \{v \in X, \exists a : G \to A, a$ is EFX in $(A, G, v)\}$.

Lemma 11. *If E is open in X, then $E = X$.*

Proof.

For $\mathfrak{a} : G \to A$, let $E_a = \{v \in X, a$ is EFX in $(A, G, v)\}$. We have $E_a = \bigcup_{i,j \in A} \{v \in X, B_j(\mathfrak{a}) \neq \varnothing \to v_i(B_\mathfrak{a}(j)) - \min_{v_i} B_\mathfrak{a}(j) \leq v_i(B_\mathfrak{a}(i))\}$. As, for $S \subset G, S \neq \varnothing$ and $i \in A$, $v \mapsto \min_{v_i} B_\mathfrak{a}(j)$ is continuous, E_a is a closed set.

Furthermore, $G \to A$ is finite, and thus E is closed. By assumption, E is open. As X is connected, we have $E \in \{\varnothing, X\}$. We know that $E \neg \varnothing$, because EFX allocations always exists for identical valuations. Thus we can conclude $E = X$.

Proposition 2. *If, for $v \in E$, $i \in A$ and $S \subset G$, there exists $\varepsilon \in \mathbb{R}^{+*}$ such that for $v' \in X$, if v' differs from v only on i and S and $|v'_i(S) - v_i(S)| \leq \varepsilon$, then $v' \in E$, then $E = X$.*

Proof.

Let ε be the smallest value considered in the above condition. Then it is easy to see that, for $v \in E$, $B_\infty(v, \varepsilon) \subset E$ by changing all utilities one by one. Thus E is open and by the preceding lemma, $E = X$.

Note that a similar argument can be made restricting to additive valuations, considering only the $v_i(\{g\}), i \in A, g \in G$.

6.4 Probabilistic method

So far, we have only seen positive arguments w.r.t. the existence of EFX allocations. Here, we show a negative one by giving a lower bound on the probability that some strong envy edge appears in a random allocation. We consider (A, G, v) an additive and non-degenerate instance, (Ω, Π, P) a probalistic space and $\mathfrak{a} : \Omega \to G \to A$ a random variable with uniform distribution:

$$\forall i \in A, \forall g \in G, P(\mathfrak{a}(g) = i) = \frac{1}{|A|}$$

Lemma 12. *Let $i \neq j \in A$. Then $\frac{1}{2|A|} \leq P(2 \leq envy_{\mathfrak{a}}(i, j))$.*

Proof.

Let $A = \{g_1, ..., g_n\}$ with $(v_i(g_k))_{k \in [\![1,n]\!]}$ increasing. Then:

$$\begin{aligned}
P(2 \leq envy_{\mathfrak{a}}(i,j)) &= P(v_i(B_{\mathfrak{a}}(i)) < v_i(B_{\mathfrak{a}}(j)) - \inf_{B_{\mathfrak{a}}(j)} v_i) \\
&= \sum_{k=1}^{n} P(v_i(B_{\mathfrak{a}}(i)) < v_i(B_{\mathfrak{a}}(j)) - v_i(g_k) | \underset{B_{\mathfrak{a}}(j)}{argmin}\ v_i = g_k) P(\underset{B_{\mathfrak{a}}(j)}{argmin}\ v_i = g_k) \\
&= \sum_{k=1}^{n} P(\sum_{p=1}^{n} v_i(g_p) \mathbb{1}_{\mathfrak{a}(g_p)=i} < \sum_{p=k+1}^{n} v_i(g_p) \mathbb{1}_{\mathfrak{a}(g_p)=j} | \underset{B_{\mathfrak{a}}(j)}{argmin}\ v_i = g_k) \tfrac{1}{n} (1 - \tfrac{1}{n})^{k-1} \\
&= \tfrac{1}{n} \sum_{k=1}^{n} P(\sum_{p=1}^{k-1} v_i(g_p) \mathbb{1}_{\mathfrak{a}(g_p)=i} < \sum_{p=k+1}^{n} v_i(g_p)(\mathbb{1}_{\mathfrak{a}(g_p)=j} - \mathbb{1}_{\mathfrak{a}(g_p)=i}) | \underset{B_{\mathfrak{a}}(j)}{argmin}\ v_i = g_k)(1 - \tfrac{1}{n})^{k-1} \\
&\geq \tfrac{1}{n} P(0 < \sum_{p=1}^{n} v_i(g_p)(\mathbb{1}_{\mathfrak{a}(g_p)=j} - \mathbb{1}_{\mathfrak{a}(g_p)=i}) | \underset{B_{\mathfrak{a}}(j)}{argmin}\ v_i = g_0) \\
&= \tfrac{1}{n} P(0 < \sum_{p=1}^{n} v_i(g_p)(\mathbb{1}_{\mathfrak{a}(g_p)=j} - \mathbb{1}_{\mathfrak{a}(g_p)=i}) \\
&= \tfrac{1}{2n}
\end{aligned}$$

The last equality comes from the fact that, $\sum_{p=1}^{n} v_i(g_p)(\mathbb{1}_{\mathfrak{a}(g_p)=j} - \mathbb{1}_{\mathfrak{a}(g_p)=i})$ is symmetric and can not take the value 0, because $B_{\mathfrak{a}}(i) \neq B_{\mathfrak{a}}(j)$ and the instance is non-degenerate.

7 Operational calculus algebra

It has occured at several moments during the internship that we wanted to solve recurrence formulas. Some of them (and especially the one-dimensional linear recurrences) can be handeled easily using the operational calculus algebra introduced by Berg in [Ber67]. This algebra can be generalised to the multi-dimensional case as follows:

Let $\mathbb{K}$ be a field and $n \in \mathbb{Z}_{\geq 0}$.

Definition. The field of operators over $\mathbb{K}$ of dimension n is

$$\mathcal{O}_n(\mathbb{K}) = \Big(\{x \in \mathbb{K}^{\mathbb{Z}^n}, \exists k \in \mathbb{Z}, supp(x) \subset [\![k, +\infty[\![\}^n, +, \cdot\Big)$$

with + the standard addition and :

$$\cdot : (a,b) \mapsto (\sum_{\delta \in 2^n} (-1)^{\sum_{i=1}^{n} \delta_i} \sum_{\substack{p,q \in \mathbb{Z}^I \\ \forall 1 \le i \le n, p_i + q_i = k_i - \delta_i}} a_p b_q)_{k \in (\mathbb{Z}_{\ge 0})^n})$$

Lemma 13. *$\mathcal{O}_n(\mathbb{K})$ is a field with neutral elements $(0)_{p \in \mathbb{Z}^I}$ and $(\mathbb{1}_{(\mathbb{Z}_{\ge 0})^I}(p))_{p \in \mathbb{Z}^I}$. Furthermore, for $n, m \in \mathbb{Z}_{\ge 0}$, $\mathcal{O}_{n+m}(\mathbb{K})$ and $\mathcal{O}_n(\mathcal{O}_m(\mathbb{K}))$ are isomorphic.*

Proof.

The fact that $\mathcal{O}_n(\mathbb{K})$ is a ring can be seen directly. For $n, m \in \mathbb{Z}_{\ge 0}$, the isomorphism from $\mathcal{O}_n(\mathcal{O}_m(\mathbb{K}))$ to $\mathcal{O}_{n+m}(\mathbb{K})$ is given by the following formula (with $k = p :: q$ the decomposition of $k \in \mathbb{Z}^{n+m}$ into $p \in \mathbb{Z}^n$ and $q \in \mathbb{Z}^m$):

$$u \mapsto ((u_q)_p)_{k = p::q \in \mathbb{Z}^{n+m}}$$

Finally, the field property is immediate by induction with the case $n = 1$ already known and using the preceding isomorphism.

Lemma 14. *$x \mapsto (x\mathbb{1}_{\mathbb{Z}^n_{\ge 0}}(p))_{p \in \mathbb{Z}^n}$ is an embedding from $\mathbb{K}$ to $\mathcal{O}_n(\mathbb{K})$.*

The preceding lemma shows that $\mathcal{O}_n(\mathbb{K})$ can be seen as an algebra over $\mathbb{K}$.
There exists an equivalent formula for the product in $\mathcal{O}_1(\mathbb{K})$ that is easier to use:

Lemma 15. *Let $u, v \in \mathcal{O}_1(A)$. Then :*

$$uv = (\sum_{\substack{p,q \in \mathbb{Z} \\ p+q=n}} u_p(v_q - v_{q-1}))_{n \in \mathbb{Z}}$$

The main tools in the operational algebra are the shift operators. Let, for $i \in [\![1, n]\!], q_i = 1 + (\delta_{-1,p_i})_{p \in \mathbb{Z}^n}$.

Lemma 16. *Let $u \in \mathcal{O}_n(A)$ and $i \in I$. We have :*

$$q_i u = (u_{p+\mathbb{1}_{\{i\}}})_{p \in \mathbb{Z}^n_{\ge 0}}$$

8 Experiments

The majority of the examples found during the internship came from tests of random allocations. The code is written in C++, but except for templates, everything is in C-style. Due to space constaints, it has only been possible to handle additive valuations.

The project decomposes into six modules. All modules use the same model for representing instances and allocations. Two constants n and m give the number of agents and the number of goods. An instance is then simply an $n \times m$ array of integers. At some point, it has been considered to use floating points numbers, especially for greedy algorithms, but rounding issues made most of the algorithms break and thus the testing unefficient. The model for allocation is the same as in this report: an array of size m giving each good's owner. As they ended up being useful in several modules, the envy graph and the champions graph have been included in the model. All these objects have been defined as global variables, as there were to be accessed almost everywhere in the code. The initialization of the instance, i.e. the allocation of the memory, has been hided in a macro taking only n and m as arguments.

Above the hierarchy of modules lie a few utility functions, i.e. functions that are useful in several modules. This includes general utility functions, like print utilities, sort utilities (in particular a function to compute the permutation that sorts an array). The other utilities are more specific to the fair allocation problem. This includes function for computing the value of a bundle, envy and champions graph computation and some fairness criteria. The only utility that I did not write and is not in the standard library is an implementation of the hungarian algorithm [Pay17].

The first module implemented concerns the algorithm for three agents. Towards a generalization of the algorithm, an envy-cycle elimination routine was first implemented together with the result in lemma 3. Surprisingly, these two update rules solve already a significant fraction of the instances.This basis has then been used to test various update rules and find instances where they were not sufficient.

The most useful and technically involved module generates all allocations given an instance. The enumerating procedure has the following signature:

```
void enumerate(bool (*filter)(), void (*action)(), int (*cut)(), bool *change,
        int base = 0);
```

Note that everything is coded with reusability in mind. The filter and action functions control which allocations we are interested in and what we do with them. Most of the time, the action is simply setting a flag to remember that an allocation has been found, counting the number of allocations, or printing the found allocations. The cut function allows, when possible, to use some sort of backtracking. The allocations are generated in the inverse lexicographic order. So if one can decide easily that something in the rightmost part of the allocation make it impossible for the allocation to pass the filter, a whole branch of the exploration tree can be cut. For example, when looking for Pareto-optimal allocation, if two agents can exchange two goods and both increase their valuation, we can cut at the leftmost good, because while these two goods are allocated the way they are, the allocation can not be PO. The change and base arguments are used to control which good the algorithm should actually allocate and whether they are allowed to be left unallocated.

The four remaining modules have proven to be less useful. They were used to explore some greedy algorithms, the decreasing method, the partitions and pairs of graphs verifying some conditions. The partition module works more or less the same way as the allocations generator module, the main difference being the generating procedure that iterates other partitions of $[\![1, m]\!]$ into n disjoint sets, using an algorithm from [Knu86].

9 Conclusion

This internship has been my first real research experience, in the sense of studying deeply a precise subject and try to find ideas to solve open problems. One of the most difficult challenged I have encountered is most probably the language. It has been quite difficult to exchange with the members of the MPI at the beginning. My english has without any doubt much improved. This internship

has also been the occasion to test my programming skills, as I have had to produce quickly efficient ways of testing ideas.

I would like to thank the MPI and in particular the D1 for their welcome. Many thanks to those I worked closely with, namely Bhaskhar Ray Chaudhury, Hannaneh Akrami, Golnoosh Shakharami and especially my supervisor Kurt Mehlhorn.

References

[ABF+20] Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, Alexandros Hollender, and Alexandros A. Voudouris. Maximum nash welfare and other stories about EFX. *CoRR*, abs/2001.09838, 2020.

[AGSS16] Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. Nash social welfare, matrix permanent, and stable polynomials. *CoRR*, abs/1609.07056, 2016.

[ANM19] Georgios Amanatidis, Apostolos Ntokos, and Evangelos Markakis. Multiple birds with one stone: Beating 1/2 for EFX and GMMS via envy cycle elimination. *CoRR*, abs/1909.07650, 2019.

[BDN+19] Johannes Brustle, Jack Dippel, Vishnu V. Narayan, Mashbat Suzuki, and Adrian Vetta. One dollar each eliminates envy. *CoRR*, abs/1912.02797, 2019.

[BEF20] Moshe Babaioff, Tomer Ezra, and Uriel Feige. Fair and truthful mechanisms for dichotomous valuations. *CoRR*, abs/2002.10704, 2020.

[Ber67] Lothar Berg. *Introduction to the Operational Calculus*, volume 2. North-Holland Publishing Co., Amsterdam, 1967.

[BL14] Sylvain Bouveret and Michel Lemaître. Characterizing conflicts in fair division of indivisible goods using a scale of criteria. In Ana L. C. Bazzan, Michael N. Huhns, Alessio Lomuscio, and Paul Scerri, editors, *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pages 1321–1328. IFAAMAS/ACM, 2014.

[BLLW20] Xiaohui Bei, Shengxin Liu, Xinhang Lu, and Hongao Wang. Maximin fairness with mixed divisible and indivisible goods. *CoRR*, abs/2002.05245, 2020.

[BM17] Siddharth Barman and Sanath Kumar Krishna Murthy. Approximation algorithms for maximin fair division. *CoRR*, abs/1703.01851, 2017.

[BMV17] Siddharth Barman, Sanath Kumar Krishna Murthy, and Rohit Vaish. Finding fair and efficient allocations. *CoRR*, abs/1707.04731, 2017.

[BMV18] Siddharth Barman, Sanath Kumar Krishna Murthy, and Rohit Vaish. Greedy algorithms for maximizing nash social welfare. *CoRR*, abs/1801.09046, 2018.

[BT03] Graham R. Brightwell and Prasad Tetali. The number of linear extensions of the boolean lattice. *Order*, 20(4):333–345, 2003.

[CCK09] Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On allocating goods to maximize fairness. *CoRR*, abs/0901.0205, 2009.

[CG18] Richard Cole and Vasilis Gkatzelis. Approximating the nash social welfare with indivisible items. *SIAM J. Comput.*, 47(3):1211–1236, 2018.

[CGH19] Ioannis Caragiannis, Nick Gravin, and Xin Huang. Envy-freeness up to any item with high nash welfare: The virtue of donating items. *CoRR*, abs/1902.04319, 2019.

[CGM20] Bhaskar Ray Chaudhury, Jugal Garg, and Kurt Mehlhorn. EFX exists for three agents. *CoRR*, abs/2002.05119, 2020.

[CI20] Ioannis Caragiannis and Stavros Ioannidis. Computing envy-freeable allocations with limited subsidies. *CoRR*, abs/2002.02789, 2020.

[CKM+16] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. In Vincent Conitzer, Dirk Bergemann, and Yiling Chen, editors, *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 305–322. ACM, 2016.

[CKMS19] Bhaskar Ray Chaudhury, Telikepalli Kavitha, Kurt Mehlhorn, and Alkmini Sgouritsa. A little charity guarantees almost envy-freeness. *CoRR*, abs/1907.04596, 2019.

[FSVX20] Rupert Freeman, Sujoy Sikdar, Rohit Vaish, and Lirong Xia. Equitable allocations of indivisible chores. *CoRR*, abs/2002.11504, 2020.

[GKK19] Jugal Garg, Pooja Kulkarni, and Rucha Kulkarni. Approximating nash social welfare under submodular valuations through (un)matchings. *CoRR*, abs/1912.12541, 2019.

[GMT19] Jugal Garg, Peter McGlaughlin, and Setareh Taki. Approximating maximin share allocations. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, volume 69 of *OASICS*, pages 20:1–20:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[GT19] Jugal Garg and Setareh Taki. An improved approximation algorithm for maximin shares. *CoRR*, abs/1903.00029, 2019.

[HL19] Xin Huang and Pinyan Lu. An algorithmic framework for approximating maximin share allocation of chores. *CoRR*, abs/1907.04505, 2019.

[HPPS20] Daniel Halpern, Ariel D. Procaccia, Alexandros Psomas, and Nisarg Shah. Fair division with binary valuations: One rule to rule them all. Technical report, 2020.

[KKM19] Andreas Karrenbauer, Pavel Kolev, and Kurt Mehlhorn. Convergence of the non-uniform physarum dynamics. *CoRR*, abs/1901.07231, 2019.

[Knu86] Donald E. Knuth. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1.* Addison-Wesley, 1986.

[KPW18] David Kurokawa, Ariel D. Procaccia, and Junxing Wang. Fair enough: Guaranteeing approximate maximin shares. *J. ACM*, 65(2):8:1–8:27, 2018.

[MS18] Pasin Manurangsi and Warut Suksompong. When do envy-free allocations exist? *CoRR*, abs/1811.01630, 2018.

[Pay17] James Payor. "weighted-bipartite-perfect-matching" project, 2017. https://github.com/jamespayor/weighted-bipartite-perfect-matching.

[PR17a] Benjamin Plaut and Tim Roughgarden. Almost envy-freeness with general valuations. *CoRR*, abs/1707.04769, 2017.

[PR17b] Benjamin Plaut and Tim Roughgarden. Communication complexity of discrete fair division. *CoRR*, abs/1711.04066, 2017.

[SAH17] Erel Segal-Halevi, Haris Aziz, and Avinatan Hassidim. Fair allocation based on diminishing differences. *CoRR*, abs/1705.07993, 2017.

[SK87] Jichang Sha and Daniel J. Kleitman. The number of linear extensions of subset ordering. *Discret. Math.*, 63(2-3):271–278, 1987.

[TC13] Prasad Tetali Teena Carroll, Joshua Cooper. Counting antichains and linear extensions in generalizations of the boolean lattice. http://people.math.gatech.edu/ tetali/PUBLIS/CCT.pdf. Preprint., 2013.