EFX Allocations on Graphs

George Christodoulou[*] Amos Fiat[†] Elias Koutsoupias[‡] Alkmini Sgouritsa[§]

July 24, 2023

Abstract

We study *envy freeness up to any good (EFX)* in settings where valuations can be represented via a graph of arbitrary size where vertices correspond to agents and edges to items. An item (edge) has zero marginal value to all agents (vertices) not incident to the edge. Each vertex may have an arbitrary monotone valuation on the set of incident edges. We first consider allocations that correspond to orientations of the edges, where we show that EFX does not always exist, and furthermore that it is NP-complete to decide whether an EFX orientation exists. Our main result is that (EFX) allocations exist for this setting. This is one of the few cases where EFX allocations are known to exist for more than 3 agents.

1 Introduction

We consider a setting where m indivisible goods need to be allocated, in a *fair* manner, among n agents. It is well-known that standard fairness notions that apply to the case of divisible goods, like envy-freeness, where every agent prefers their own bundle over any bundle given to some other agent, do not always exist.

Several variants of envy freeness have been considered in the literature. One such notion is that of *envy freeness up to one good* (EF1) [Budish(2011)] which is an allocation of m indivisible goods to n agents, $X_1, \dots, X_n$ so that for every pair of agents i, j, there exists an item $x \in X_j$ such that agent i does not prefer $X_j \setminus \{x\}$ over X_i. Such allocations are known to always exist [Lipton et al.(2004)].

A stronger requirement is that of *envy freeness up to any good (EFX)* [Caragiannis et al.(2019)]. An EFX allocation is a partition of m indivisible goods to n agents, $X_1, \dots, X_n$ so that for every pair of agents i, j, agent i does not prefer $X_j \setminus \{x\}$ over X_i, *for all* items $x \in X_j$. Clearly, every EFX allocation is also EF1.

It is a major open question whether there is always an EFX allocation; to quote Ariel Procaccia, it is "Fair Division's Most Enigmatic Question" [Procaccia(2020)]. Despite much effort, EFX is only known to exist for two agents for general valuations [Plaut and Roughgarden(2020)], and for three agents when the valuations are additive[1] [Chaudhury et al.(2020)][2].

For the case of multiple agents, EFX is only known to exist for special cases and various valuation function restrictions: plaut2020almost show that EFX allocations exist if all agents have the *same* valuation function. Mahara21 extended this result for the setting where every agent has one of two valuation functions (can be arbitrary valuation functions). Regarding restricted classes of valuations, all of the following have EFX allocations: GMV23 considered different additive valuations, but for at most two types of items, Amanatidis21 considered valuation domains where there are only two distinct possible values for each good, BabaioffEF21 considered submodular *dichotomous* valuations, in which the marginal value of any item to a set is either 0 or 1, while HosseiniSVX21 considered *lexicographic preferences*. Finally, it is also known that EFX allocations exist for arbitrary allocations if the number of goods is at most 3 more than the number of agents ($m \leq n + 3$) [Mahara(2021)].

In this paper, we consider a scenario where the number of agents "with interest" in any specific item is limited. Concretely, we prove that EFX allocations exist with multiple players, for a class of

[*]Aristotle University of Thessaloniki and Archimedes/RC Athena, Greece. Email: `gichristo@csd.auth.gr`

[†]Tel-Aviv University, Israel. Email: `fiat@tau.ac.il`

[‡]University of Oxford, UK. Email: `elias@cs.ox.ac.uk`

[§]Athens University of Economics and Business and Archimedes/RC Athena, Greece. Email: `alkmini@liv.ac.uk`

[1]In additive valuations, the value of an agent for a bundle of items is equal to the sum of the individual values of the items in the bundle.

[2]In fact, more recently the result was generalized to capture the case where only one of the agents is restricted to having an additive valuation but the other two may have arbitrary valuations [Akrami et al.(2022)].

valuation functions associated with a graph. In this graph, items are associated with edges and agents with vertices. An item x is said to be irrelevant for agent i if for all sets of items S we have that $v_i(S \cup \{x\}) = v_i(S)$ (an item that is not irrelevant is said to be relevant). In our model, an item (edge) is irrelevant for all agents that are not endpoints of the edge. *I.e.*, for any edge $e = (i, j)$ and for any set of items (edges) S, $v_k(S \cup \{e\}) = v_k(S)$ for all agents $k \neq i, j$. For the endpoints i and j all we require is that the valuations are monotone but *otherwise unrestricted.* Monotonicity means that the marginal values $v_i(S \cup \{e\}) - v_i(S)$ and $v_j(S \cup \{e\}) - v_j(S)$ are nonnegative. Unrestricted values mean that the valuation function for agent i (and j) is an arbitrary function of the adjacent edges.

One direct motivation of the graph-based approach is the allocation of resources or goods in a geographic setting. Agents are interested only in the resources close by, are uninterested in further locations, and we seek a complete and fair allocation. Examples include office space/lecture halls in academic environments, "areas of influence" of neighboring powers such as Persia, Athens, and the Peloponnesian league in the 5th century BCE, the reorganization of Europe at the congress of Vienna, the areas of influence set at Yalta at the end of WWII, etc.

To avoid confusion we remark that graphs have been used to describe yet another EFX relaxation [Payan et al.(2022)]. In the context of [Payan et al.(2022)] vertices are agents (as in our setting) but an edge is not an item but rather represents a requirement that the EFX condition (or other fairness desiderata) need hold only for endpoints of edges. This is a model that tries to formalize the intuition that one can only envy neighbors.

1.1 Our Contributions

We propose the study of EFX in graphs where vertices correspond to agents and edges to goods. An allocation of particular interest is an *orientation* which is an assignment of every item to one of the two endpoints. Orientations are desirable as they allocate the goods only to interested parties.

We show the following:

- The question of whether a graph has an EFX orientation is hard (*NP*-complete) — in general EFX orientations need not exist, even in very simple settings.
- There is always an EFX allocation for graphs. In particular this implies that there are cases where any EFX allocation must assign items to agents for which they are irrelevant.

We remark that many of the major problems in algorithmic game theory, such as the complexity of Nash equilbria[Daskalakis et al.(2009), Chen et al.(2009)] and truthful scheduling (Nisan-Ronen conjecture [Nisan and Ronen(2001), Christodoulou et al.(2022), Christodoulou et al.(2023)]), retain their characteristics and difficulty when restricted to graph settings. This raises the hope that one can make progress on the fundamental EFX problem by considering extensions to the graph model proposed in this paper, see Section 5.

1.2 Further Related Work

There is extensive literature on the area of fair allocation of indivisible items. In this section we discuss work closely related to EFX and we refer the reader to [Amanatidis et al.(2022)] for other fairness concepts.

Relaxations There have been several studies of weaker notions of EFX and conditions under which they are known to exist. plaut2020almost considered approximate EFX showing that there is always a 1/2-EFX, Chan00W19 showed that it can be computed in polynomial time, while AMN20 improved the approximation guarantee to $1/\phi \approx 0.618$ which is currently the best known. AkramiRS22 considered EFkX, which is envy-freeness up to *any k goods*, and showed existence for $k = 2$ for restricted additive valuations. Finally epistemic defined and established existence for *epistemic* EFX where for every agent i, there is a way to shuffle the goods of the other agents such that agent i does not envy any other agent up to any good.

EFX with Charity CGH19 initiated the research of finding EFX allocations that satisfy certain properties where some goods remain unallocated and are rather donated to a "charity". CGH19 guarantee that in the outcome, each agent is allocated a set of goods that they value at least half as much as they value their allocated goods in the outcome that maximizes the Nash social welfare, an important, well-known notion of fairness. A restricted charity was studied in [Chaudhury et al.(2021b)], where the number of goods given to charity is at most the number of agents and no agent envies the charity. Follow-up works reduce the number of charity goods [Chaudhury et al.(2021a), Berendsohn et al.(2022), Akrami et al.(2022), Jahan et al.(2022)] and the state-of-the-art for four agents is to give at most one good to charity [Berger et al.(2022)].

2 Model and Preliminaries

There are n agents labeled $1, \dots, n$. Let E, $|E| = m$, be a set of items. For all $i = 1, \dots, n$ let $v_i : 2^E \to \Re^+$ be such that $v_i(X)$ is the value agent i has for the subset of items $X \subseteq E$. The valuation functions are considered monotone.

A full/complete allocation (or just an allocation) is a partition of E items amongst the n agents, where agent i gets the set X_i. A partial allocation is an allocation of a subset of items in E.

Given an allocation or partial allocation $X_1, \dots, X_n$, agent i is said to *envy* agent j if $v_i(X_i) < v_i(X_j)$, agent i is said to *strongly envy* agent j if for some $x \in X_j$ $v_i(X_i) < v_i(X_j \setminus \{x\})$. Similarly, agent i is said to envy a set of items X if $v_i(X_i) < v_i(X)$, and agent i is said to strongly envy a set of items X if for some $x \in X$ $v_i(X_i) < v_i(X \setminus \{x\})$.

An EFX allocation is an allocation $X_1, \dots, X_n$ so that no agent strongly envies another: for all $1 \le i, j \le n$

$$v_i(X_i) \ge v_i(X_j \setminus \{x\}), \forall x \in X_j.$$

Similarly we define an EFX partial allocation, where some items may be unallocated and ignored.

An item $e \in E$ is said to be *irrelevant* for agent k,if

$$v_k(X \cup \{e\}) = v_k(X), \forall X \subseteq E.$$

If an item is not irrelevant for agent k is said to be relevant for agent k. We consider valuations with a special structure where every item is relevant to at most 2 agents and for every pair of agents there exists at most one item relevant to both of them. This class of valuations can be represented via a graph $G = (V, E)$ where the vertices represent the agents, i.e., $V = \{1, \dots, n\}$, and the set of edges E is the set of items, where the item represented by the edge (i, j) is only relevant to both agents i and j.

Given a graph $G = (V, E)$, an *orientation* is an allocation with the additional property that every edge is assigned to one of the two endpoints. This can be represented by directing the edges of the graph towards the vertex receiving the edge. We call such a directed graph the *orientation graph*. A partial allocation $\mathbf{X}$ is an orientation if all allocated edges are assigned to one of the two endpoints. The orientation graph is then defined by keeping only the assigned edges. An (partial) orientation is said to be an EFX orientation if the resulting (partial) allocation is an EFX allocation.

3 EFX orientations

An orientation is an assignment where every edge (item) is assigned to one of the two endpoints. Note that every orientation is automatically an EF1 allocation as no agent has more than one relevant item allocated to any other agent.

Definition 1. A graph instance is called symmetric if for every edge $e = (i, j)$ the values of both agents are the same, i.e., $w_i(e) = w_j(e)$. We refer to this common value as the weight of that edge.

Unfortunately, the following simple example demonstrates that there is not always an EFX orientation, even for the case of 4 agents and for a symmetric instance.

Example 1. *Consider a symmetric graph instance with* 4 *agents. Edges* $(1, 2)$ *and* $(3, 4)$ *are heavy with weight equal to* $v >> 1$. *All other edges are light with weight equal to* 1. *Observe that no matter how one orients the heavy edges, the endpoint that receives the edge will be envied by the other endpoint. Note that any orientation of the light edge between the agents that receive the heavy edges, violates EFX. This is because the recipient of that edge is also recipient of a heavy edge, hence she will be envied (by the corresponding endpoint of the heavy edge), even if the light edge is removed.*

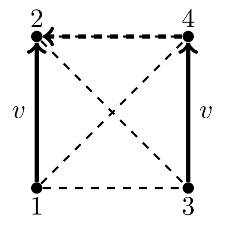

Figure 1: The weight of heavy edges is $v \gg 1$ for both agents and they are represented by thick lines. The light edges carry low weight (with value 1 for both agents) and they are represented by dashed lines. No matter how the two thick edges are oriented, this will create envy. In the figure, agents 2 and 4 receive the thick edges. Note that one of them needs to also receive the light edge $(2,4)$. In the figure, it is given to agent 2. Now observe that irrespectively of the orientation of the rest of the edges, agent 1 envies agent 2 (even if edge (2,4) is removed), so EFX is violated.

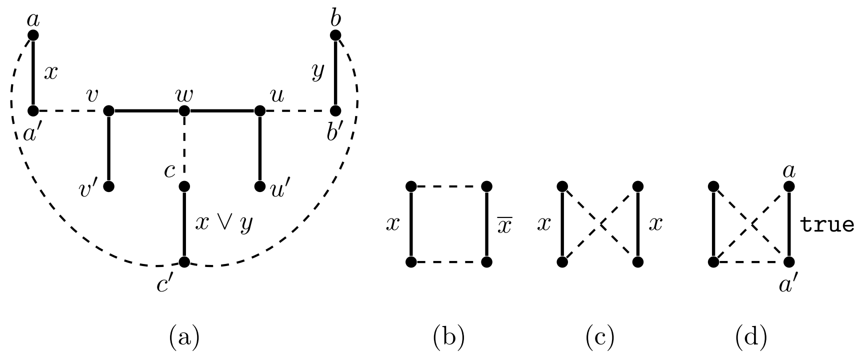

Figure 2: In these figures, heavy edges (with some fixed high value $v \gg 1$ for both agents) are represented by thick lines and light edges (with value 1 for both agents) are represented by dashed lines. `true` corresponds to orienting an edge towards its upper node. (a) OR gadget: edge $x \vee y$ is given to agent c if and only if edge x is given to agent a or edge y is given to agent b. Furthermore, there is an EFX allocation for every orientation of edges x and y. (b) NOT gadget. (c) consistency gadget. (d) `true` gadget: in every EFX orientation the edge (a, a') must be oriented towards a.

The example above excludes the possibility that there is always an EFX orientation[3]. It will be desirable though to be able to recognize efficiently graph instances that attain an EFX orientation. Theorem 2 below shows that deciding whether a given graph instance has an EFX orientation is NP-complete.

Theorem 2. *It is an NP complete problem to decide whether a graph has an EFX orientation. This remains true even for symmetric graphs.*

Proof. We show a reduction from circuit satisfiability to the EFX orientation problem. For this, we consider symmetric graphs with two types of edges, heavy edges with value some fixed $v \gg 1$ and light edges with value 1.

We show how to simulate the elements of a Boolean circuit. It suffices to do this for the following functions: $f(x) = \texttt{true}$, $f(x) = \overline{x}$, $f(x, y) = x \vee y$ and to show how we can consistently create copies of a literal.

Given a Boolean circuit, each literal is represented by a fixed heavy edge and its values (true or false) correspond to its two orientations. Figure 2 shows the gadgets of the reduction.

The most important gadget that captures the OR gate is given in Figure 2(a). The subgraph captures $x \vee y$, where true values for edges x, y, and $x \vee y$ correspond to allocating them to the upper node. More precisely, *in every EFX allocation edge $x \vee y$ is given to agent c if and only if edge x is given to agent a or edge y is given to agent b.*

To see this notice first that if edge x is oriented towards the upper node a (i.e., literal x is true), edge $x \vee y$ must be also oriented towards the upper node c. This is so, because otherwise, no matter how edge (a, c') is oriented, some of the agents a' or c will strongly envy agents a or c', respectively. By symmetry

[3]The reader can verify that there is however, an EFX allocation.

the same holds for edge y, so we get that if any of the edges x and y are oriented towards their upper node, then $x \vee y$ must be oriented towards its upper node in an EFX orientation.

We also show that if x is oriented towards its upper node, no matter what happens with y, there is an EFX orientation for all edges of this subgraph. We have already argued that (a, c') and (c, c') are oriented towards c' and c, respectively. For the heavy edges in the middle, we orient the edges so that we have the directed path (v', v, w, u, u'), i.e., edges $(v'v,), (v, w), (w, u), (u, u')$ are oriented towards v, w, u, u', respectively. We further orient edge (a', v) towards a'. Up to this point each vertex is assigned at most one edge and therefore EFX is satisfied. Orienting (w, c) towards w doesn't violate EFX, even if w also receives the heavy edge (v, w), because v also receives the heavy edge (v', v). The only remaining edges to orient are $(u, b'), (b, c')$ which we orient according to the orientation of (b, b') in order to create a directed path (u, b', b, c') or (c', b, b', u). More specifically, if (b, b') is oriented towards b, we orient (u, b') and (b, c') towards b' and c', respectively, and it is easy to see that the final orientation satisfies EFX. If (b, b') is oriented towards b', we orient (u, b') and (b, c') towards u and b, respectively; note that u is not strongly envied by w, as w receives the heavy edge (v, w). Therefore, there is always an EFX orientation.

By symmetry there is an EFX orientation for all edges of this subgraph when y is oriented towards its upper node, no matter what happens with x.

We finally consider the case where both edges x and y are oriented towards their lower node. In an EFX orientation, edges (a', v) and (b', u) are oriented towards v and u respectively. This in turns means that (v, v') and (u, u') are oriented towards v' and u', respectively. The edges (v, w) and (u, w) cannot both be oriented towards w, since v and u would strongly envy w. On the other hand, if neither edge was allocated to w, w would strongly envy v and u. Therefore w receives only one heavy edge. W.l.o.g., (v, w) and (u, w) are oriented towards w and u, respectively. Then, (w, c) is oriented towards c, otherwise v would strongly envy w. Finally edge (c, c') is oriented towards c', otherwise c' would strongly envy c, and (a, c') and (b, c') are oriented towards a and b, respectively. In summary, if both edges x and y are oriented towards their lower node (i.e., if both literals x and y are false) then edge $x \vee y$ must be oriented towards it lower node (i.e., $x \vee y$ is false).

Similarly, the other gadgets in Figure 2 show how to implement the other elements of a Boolean circuit with OR and NOT gates. Putting such gadgets together, it is straightforward to simulate any Boolean circuit with OR and NOT gates, which shows that deciding whether a graph has an EFX orientation is NP-complete. □

4 An EFX allocation exists for all graphs

In this section we prove the following theorem that a complete EFX allocation always exists in any graph (and can be found in polynomial time). We note that this allocation is not necessarily an orientation.

Theorem 3. *There is a complete EFX allocation for any graph that can be found in polynomial time.*

We prove this theorem via several lemmata and claims. We first give some definitions.

4.1 Definitions

Given a partial allocation $\mathbf{X}$, we define the following:

- For any vertex i, $U_i(\mathbf{X})$ is the set of edges adjacent to i that are not allocated in $\mathbf{X}$.
- For any *envied* vertex i, $S_i(\mathbf{X})$ is the set of *non-envied* vertices such that, if any of those vertex was additionally given $U_i(\mathbf{X})$, i wouldn't envy them, i.e., $j \in S_i(\mathbf{X})$ iff j is not envied and $v_i(X_i) \geq v_i(X_j \cup U_i(\mathbf{X}))$. We refer to $S_i(\mathbf{X})$ as the *safe* set of vertices for i.

Our proof of Theorem 3 is constructive by creating partial allocations with the goal to satisfy the following key properties.

Key Properties: we consider the following 4 properties of a (partial) allocation $\mathbf{X}$:

1. $\mathbf{X}$ is an EFX orientation.
2. For any vertex i and $e \in U_i(\mathbf{X})$, $v_i(X_i) \geq v_i(e)$.
3. For any envied vertex i, $v_i(X_i) \geq v_i(U_i(\mathbf{X}))$.
4. For any two envied vertices i, j, where the edge (i, j) is unallocated: $S_i(\mathbf{X}) \cap S_j(\mathbf{X}) \neq \emptyset$.

4.2 Proof of Theorem 3

For convenience we can assume without loss of generality that the graph is complete: If the graph is not complete, we can simply add extra edges of (marginal) value zero to both endpoints. An EFX allocation with respect to the complete graph gives an EFX allocation after removing the extra edges that we initially added in order to make the graph complete.

4.2.1 Proof roadmap

We use a greedy algorithm (Algorithm 1) in order to derive an initial (partial) allocation $\mathbf{X}$ where each agent receives at most one adjacent edge. We show (Lemma 4) that this allocation $\mathbf{X}$ satisfies properties (1) and (2). Those two properties are preserved in all other allocations we construct. The final goal is to find an (partial) allocation that satisfies all four properties.

An intermediate goal, when $\mathbf{X}$ *doesn't satisfy all four properties*, is to derive a (partial) allocation that satisfies properties (1) and (2) and there exists a non-envied vertex k that receives a single edge where the other endpoint is also non-envied (Lemma 5). We show this in two different cases:

- If $\mathbf{X}$ doesn't satisfy property (3), we show that after we run one round of Algorithm 2 (one run of its outer while-loop), the new allocation satisfies properties (1) and (2) and there exists a non-envied vertex k that receives a single edge where the other endpoint is also non-envied (Lemma 9).
- If $\mathbf{X}$ satisfies property (3) but not property (4), we argue about some characteristics of the allocation $\mathbf{X}$ (by using Claim 10) and we distinguish between two cases (Figures 3 and 4) regarding $\mathbf{X}$. In both cases we derive a new (partial) allocation that satisfies properties (1) and (2) and we show that there exists a non-envied vertex k that receives a single edge where the other endpoint is also non-envied (Lemma 11).

Then after running Algorithm 2, which ends when satisfying property (3), we show that k keeps receiving an edge where the other endpoint is not envied. Finally, we show that k belongs to all safe sets of the envied vertices (by using Claim 10) and therefore, property (4) is also satisfied (Lemma 12).

When all four properties are satisfied we allocate the rest of the edges as follows (Lemma 13): All unallocated edges with a non-envied endpoint are given to such an endpoint (Property (2) guarantees that the new allocation is also EFX). Each edge where both their endpoints are envied is given to a vertex that belongs to the intersection of the safe sets of those endpoints (Property (4) guarantees that the new allocation is also EFX). The final allocation is a complete EFX allocation.

4.2.2 Initial allocation

We first use the following algorithm that produces an initial (partial) orientation. We consider at least 3 agents, otherwise the problem is trivial. Roughly speaking, starting from the empty allocation, Algorithm 1 arbitrarily chooses some agent i and assigns i the most valuable adjacent edge (for i). Then, the algorithm chooses the other endpoint of that edge and assigns to it its most valuablew *available* edge, and so on until the other endpoint of the currently allocated edge has already been assigned an edge. Then, the algorithm arbitrarily chooses an agent with no assigned edge and proceeds the same way until again it finds an agent that has been already considered by the algorithm. The procedure stops when all agents have been considered.

Note that all agents will be assigned a single edge. The reason is that if we consider any agent i, he has at least 2 adjacent edges (we assumed that the graph is complete). At the time that i is considered by the algorithm it is either that all of his adjacent edges are available or some of them have already been assigned to other agents. However, only one such edge could have been assigned to another agent prior to considering agent i, because at the time that the first such edge is assigned to another agent, i is the next agent to be considered as the other endpoint of that edge. So, there will be always an available adjacent edge to be given to i.

Algorithm 1 Initial Greedy Algorithm

```
    Input: A complete graph $G$ with at least 3 vertices
    Output: $\mathbf{X}$ satisfying Properties (1) and (2) where each agent is allocated exactly one edge
 1: $N = V(G)$, $M = E(G)$
 2: while $N \neq \emptyset$ do
 3:     Choose some $i$ from $N$
 4:     while $i \in N$ do
 5:         Find $j$, such that $(i, j) \in \arg\max_e\{v_i(e) \mid e \in M\}$
 6:         $X_i = \{(i, j)\}$
 7:         $N = N \setminus \{i\}$
 8:         $M = M \setminus \{(i, j)\}$
 9:         $i \leftarrow j$
10:     end while
11: end while
```

The outcome of Algorithm 1 is an orientation that as we show in the next lemma satisfies properties (1) and (2). Moreover, there is no envy cycles, i.e., a set of agents that form a cycle where each agent envies the next agent.

Lemma 4. *Let* $\mathbf{X}$ *be the (partial) allocation derived by the greedy Algorithm 1. Then* $\mathbf{X}$ *satisfies properties* (1) *and* (2)*. Moreover the orientation graph for* $\mathbf{X}$ *has no cycles consisting entirely of envied vertices.*

Proof. Property (1) holds because $\mathbf{X}$ is clearly an orientation and since every agents receives a single edge the allocation is trivially EFX. Property (2) follows since every vertex chooses the edge that is the best current choice amongst all currently adjacent unallocated edges. Finally, consider any cycle in the orientation graph. Suppose that i is last vertex of the cycle to be assigned an edge (i, j) in Algorithm 1. i can only be envied by j, but j is assigned an edge before i, and at that point (i, j) was available, so j is assigned an edge that it (weakly) prefers to (i, j) and hence does not envy i. Therefore, no cycle of envied vertices can exist. □

We next handle the case that the allocation derived by Algorithm 1 doesn't satisfy all four properties.

4.2.3 Deriving an allocation with a special vertex k

Here we assume that the initial allocation derived by Algorithm 1 doesn't satisfy all four properties. In what follows, we show how to derive a (partial) allocation that satisfies properties (1) and (2) and there exists a non-envied vertex k that receives a single edge where the other endpoint is also non-envied. This is described in the following lemma:

Lemma 5. *If the allocation derived by Algorithm 1 doesn't satisfy all four properties, there exists an (partial) allocation that satisfies properties (1) and (2) and there exists a non-envied vertex* k *that receives a single edge where the other endpoint is also non-envied.*

For the proof of Lemma 5 we consider two different cases for the initial allocation, (i) it doesn't satisfy property (3), and (ii) it satisfies property (3) but not property (4). For those cases, we prove the statement of Lemma 5 separately in Lemma 9 for case (i) and in Lemma 11 for case (ii).

We next give a second algorithm which helps to satisfy property (3). Moreover, applying one round of the following algorithm on the initial allocation gives an allocation satisfying the statement of Lemma 5.

The high level idea of Algorithm 2 is that whenever there exists an envied agent i violating property (3), i.e., i prefers the set of the adjacent unallocated edges to what he currently has, i releases his current assignment/edge and receives that set of edges. In that way i is better off. His previously obtained edge, let's say (i, j), is given to the other endpoint j (j was the agent that envied i). j in turns releases the assignment he previously had and keeps only the edge (i, j). That way j is also better off. If j was also envied before, let's say by j', the algorithm makes a similar swapping on the assignment of j' as it did with j (so j' will only receive (j, j')), and it keeps doing those swaps until it hits a non-envied vertex.

Algorithm 2 Reducing Envy Algorithm

```
    Input: $\mathbf{X}$ satisfying Properties (1) and (2)
    Output: $\mathbf{X}$ satisfying Properties (1), (2) and (3)

1: while there exists an envied vertex $i$ s.t. $v_i(U_i(\mathbf{X})) > v_i(X_i)$ do
2:     $X_i \leftarrow U_i(\mathbf{X})$ (Edges in the previous $X_i$ become unallocated)
3:     while there exists a vertex $j$ and $e \in U_j(\mathbf{X})$ s.t. $v_j(e) > v_j(X_j)$ do
4:         $X_j \leftarrow \{e\}$ (Edges in the previous $X_j$ become unallocated)
5:     end while
6: end while
```

We first show that Algorithm 2 preserves properties (1) and (2) at each round.

Claim 6. Algorithm 2 preserves properties (1) and (2) after finishing each round of the outer while loop.

Proof. Property (2) is satisfied after each round due to the condition of the inner while loop. The allocation remains an orientation since only adjacent edges are assigned to a vertex. The allocation also remains an EFX because the only agent that changes his assignment and receives more than one edge is the first agent to be considered in each round (this is the agent considered in the outer while loop); this agent receives only unallocated edges, and since property (2) holds before the start of each round, nobody envies him. Therefore, property (1) is satisfied after the end of each round. □

In the following claims we show some properties related to Algorithm 2 that help to establish Lemma 9.

Claim 7. Any vertex receiving a new allocation during the course of Algorithm 2 cannot be envied. Additionally, no vertex previously not envied will become envied during the execution of Algorithm 2.

Proof. To see this, first note that every new allocation improves the outcome for the recipient, so if agent i doesn't prefer an assignment to another agent before, it will not prefer it after the update of i's allocation. Moreover, agents/vertices receiving a new allocation are non-envied: If vertex u is considered in the outer loop then its allocation consists of edges that were unallocated before the round start and as property (2) holds (Claim 6) no vertex prefers such a common unallocated edge to its own allocation. The allocation to u is updated in the inner loop only if the newly assigned edge to u was released by some vertex that got a better allocation, so, once again, vertex u cannot be envied. □

Claim 8. Note that any vertex k assigned an edge (k, k') in the inner loop of Algorithm 2 is not envied by any vertex and vertex k' is also not envied.

Proof. It follows from Claim 7 that vertex k cannot be envied. As property (2) holds before and after every round of the outer loop, edge (k, k') was not available previously and must have been released by some vertex k', so k' was updated and thus is not envied (again from Claim 7). □

Lemma 9. *Consider a partial allocation* $\mathbf{X}$ *satisfying properties* (1) *and* (2) *but not* (3). *After applying only one round of Algorithm 2, properties (1) and (2) are satisfied and there exists a non-envied vertex* k *such that* $X_k = \{(k, k')\}$, *where* k' *is also non-envied.*

Proof. Properties (1) and (2) are satisfied due to Claim 6. Note that there is at least one envied vertex that prefers the set of unallocated adjacent edges to its current allocation, this follows since property (3) is not satisfied. Therefore, the outer while is executed at least once, say for envied vertex k'. It follows from Claim 7 that k' is unenvied in the new allocation. Recall that k' was envied, say by vertex k (which could be envied or non-envied, and could have any number of edges allocated to it). This can only be true if k' was allocated the edge (k, k') in $\mathbf{X}$. So after k' gets the adjacent unallocated edges, the edge (k, k') is released. In the inner loop, k is assigned the edge (k, k') (and release whatever it was previously allocated) and from Claim 8 is non-envied too (k' gets a better allocation). So, after the first loop of Algorithm 2, k receives (k, k') and both k, k' are unenvied. □

We now deal with the second case where the initial allocation satisfies property (3) but not property (4). Before proceeding with Lemma 11, which is the counterpart of Lemma 5 for this case, we give the following claim which gives a connection between what an non-envied agent receives and whether it belongs to the safe set of an envied agent.

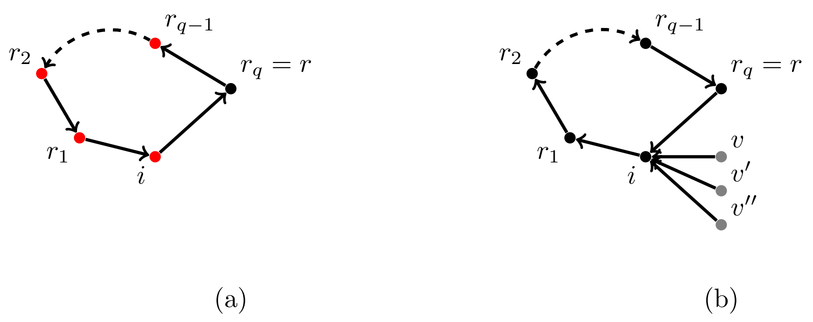

Figure 3: The figures show (part of) the orientation graphs of the allocations $\mathbf{X}$ (a) and $\mathbf{X}'$ (b) for the first scenario of the orientation graph of $\mathbf{X}$ in Lemma 11. The red vertices represent agents that are envied by some other agent, the black vertices are not envied by anyone and the gray vertices may be either envied or non-envied (may be either red or black).

Claim 10. Consider an allocation $\mathbf{X}$ where properties (1)–(3) hold and a non-envied vertex k receives an edge (k, ℓ). Then k belongs to the safe sets of all envied vertices $i \neq \ell$.

Proof. Since property (3) is satisfied we know that every envied vertex i doesn't prefer the set of the adjacent unallocated edges $U_i(\mathbf{X})$ to its allocation X_i. Every envied vertex $i \neq \ell$ has none of its adjacent edges assigned to k so adding the set $U_i(\mathbf{X})$ to the allocation of k cannot cause i to envy k and k is in the safe set $S_i(\mathbf{X})$. □

Lemma 11. *Let $\mathbf{X}$ be the partial allocation derived by the greedy Algorithm 1. Suppose that it satisfies properties (1)-(3) but not (4). We can derive a (partial) allocation $\mathbf{X}'$ satisfying properties (1) and (2) where there exists non-envied vertices k, k' such that $X'_k = \{(k, k')\}$.*

Proof. Since property (4) doesn't hold, there must be two envied vertices i and j with an unallocated edge between them such that their safe sets do not intersect, i.e., $S_i(\mathbf{X}) \cap S_j(\mathbf{X}) = \emptyset$.

Let $i = r_0$, and r_1 be the vertex that envies i. If r_1 is envied, let r_2 be the vertex that envies r_1. Continuing as above, if r_ℓ is envied let $r_{\ell+1}$ be the vertex that envies r_ℓ (this can only happen if r_ℓ receives $X_{r_\ell} = \{(r_\ell, r_{\ell+1})\}$). Since there are no cycles of only envied vertices (Lemma 4) let $r = r_q$ be the first non-envied vertex along this path. Note that $q \geq 1$ since r_q is not envied and r_0 is envied. Recall that $\mathbf{X}$ is the initial allocation where every vertex receives a single edge. By Claim 10, r is assigned (r, i) or (r, j), otherwise r belongs to both safe sets of i and j, which contradicts our assumption.

We first consider the case where r **receives the edge** (r, i) (Figure 3(a)) and therefore these vertices form a cycle in the orientation graph. Vertex j may or may not be along this cycle. In this cycle every vertex envies the previous vertex except that i does not envy r. We derive $\mathbf{X}'$ by assigning every vertex in the cycle the edge assigned to the previous vertex, and i is additionally assigned all unallocated edges adjacent to it, i.e., i receives $\{(i, r)\} \cup U_i(\mathbf{X})$ (Figure 3(b)). All vertices in the cycle but i are better off under $\mathbf{X}'$ because they receive an edge that they envied in $\mathbf{X}$. We will show in a while that i is also better off under $\mathbf{X}'$ (i prefers the new allocation to (i, r_1)). Then, it follows that all vertices in the cycle are not envied in $\mathbf{X}'$. To see that this is true for any vertex but i, consider any $r_\ell \neq i$ in the cycle that receives $(r_\ell, r_{\ell-1})$ in $\mathbf{X}'$. Agent $r_{\ell-1}$ is the only agent that could envy r_ℓ, but $r_{\ell-1}$ is better off, ergo, it prefers its allocation in $\mathbf{X}'$ to its allocation in $\mathbf{X}$ (which was $(r_\ell, r_{\ell-1})$).

We next show that i is better off and not envied under $\mathbf{X}'$:

- i prefers his new allocation $X'_i = \{(i, r)\} \cup U_i(\mathbf{X})$ to his old allocation $X_i = \{(i, r_1)\}$ because r does not belong to the safe set of i, i.e., $r \notin S_i(\mathbf{X})$. This holds since by Claim 10, r belongs to the safe set of j and if r also belonged to the safe set of i, this would violate our assumption that the two safe sets, $S_i(\mathbf{X})$ and $S_j(\mathbf{X})$, have an empty intersection.

- The only vertices that could possibly envy i in $\mathbf{X}'$ are r or a vertex v such that $(i, v) \in U_i(\mathbf{X})$. Vertex r does not envy i in $\mathbf{X}'$ since r prefers (r, r_{q-1}) to (r, i), as r_{q-1} was envied by r in $\mathbf{X}$. Moreover, i is not envied by any vertex v, such that $(i, v) \in U_i(\mathbf{X})$, because (i, v) was unallocated in $\mathbf{X}$ and by property (2), v does not prefer (i, v) to its allocation in $\mathbf{X}$, and therefore to its allocation in $\mathbf{X}'$ since no one is worse off in $\mathbf{X}'$.

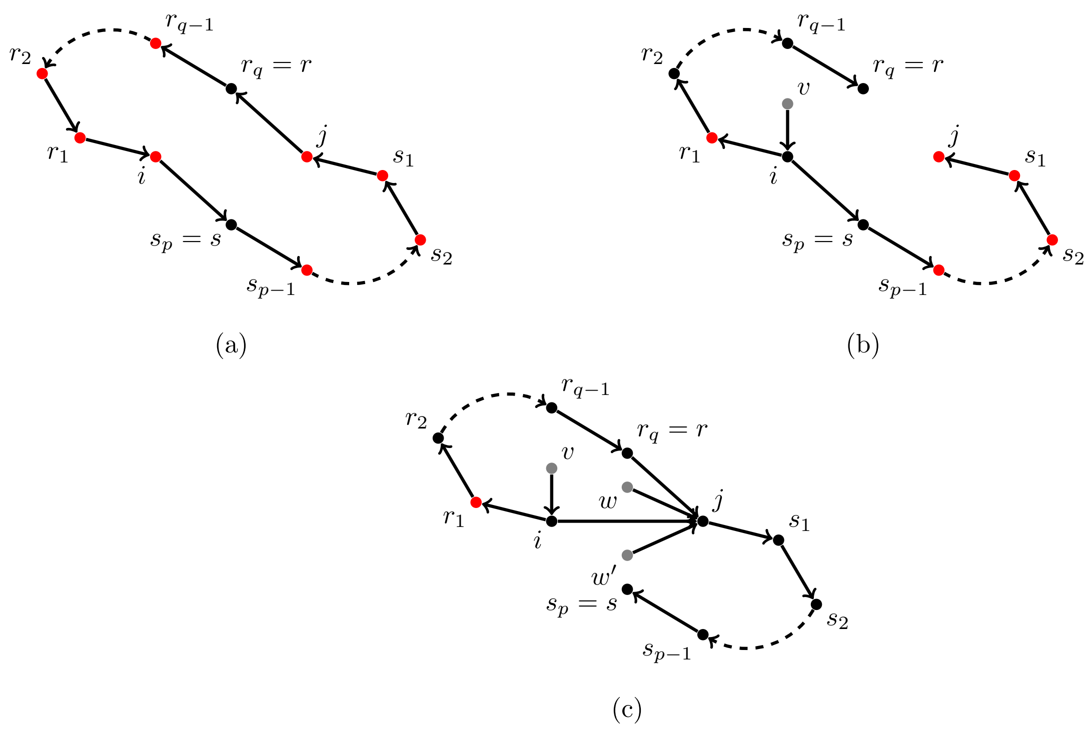

Figure 4: The figures show (part of) the orientation graphs of the allocations $\mathbf{X}$ (a) and $\mathbf{X}'$ (where (b) and (c) refer to $\mathbf{X}'$ when i doesn't prefer or prefers, respectively, (i,s) to (i,j)) for the second scenario of the orientation graph of $\mathbf{X}$ in Lemma 11. The red vertices represent agents that are envied by some other agent, the black vertices are not envied by anyone and the gray vertices may be either envied or non-envied (may be either red or black).

Since no vertex in the cycle is envied under $\mathbf{X}'$, there are two non-envied vertices k, k' (for example $k = r_q$ and $k' = r_{q-1}$) where k receives $X'_k = (k, k')$. Property (1) is satisfied since the only changes concern vertices along the cycle which are not envied and are better off in $\mathbf{X}'$. Property (2) is also satisfied because the edges allocated in $\mathbf{X}'$ are a superset of the edges allocated in $\mathbf{X}$ and all agents are better off in $\mathbf{X}'$. Therefore, the lemma follows.

We now consider the same scenario for j. Let $j = s_0, s_1, \ldots, s_p = s$ be a sequence of vertices where s_{r+1} envies s_r and s is not envied by anyone. Arguing as above, s is assigned either (s,i) or (s,j) in $\mathbf{X}$. If s is assigned (s,j), the same argument as above can be used. The vertices form a cycle in the orientation graph and we derive $\mathbf{X}'$ similarly as above by assigning every vertex in the cycle the edge assigned to the previous vertex, and j is additionally assigned $U_j(\mathbf{X})$. The lemma follows by the same arguments as above.

The one remaining scenario is that r receives $X_r = \{(r,j)\}$ and s receives $X_s = \{(s,i)\}$ (note that $r \neq s$, otherwise one of the above scenarios would apply); this case is shown in Figure 4(a). We distinguish between two cases.

- **Case 1:** *i prefers (i,j) to (i,s), or it's indifferent between the two edges.* We derive $\mathbf{X}'$ by assigning to every vertex $r_\ell \neq i$, for $\ell \in [1,q]$, the edge $(r_\ell, r_{\ell-1})$, and i is assigned its most preferred unallocated edge adjacent to it. $\mathbf{X}'$ for this case is shown in Figure 4(b). Note that the unallocated edges adjacent to i after we've done the above reallocation of all vertices $r_\ell \neq i$ are the same as the unallocated edges adjacent to i in $\mathbf{X}$. The vertex v could be j or some other vertex where the edge (i,v) is unallocated in $\mathbf{X}$.

  We note that i is not envied in $\mathbf{X}'$ because no vertex w preferred (i,v) to X_w (due to property (2) in $\mathbf{X}$) and they are not worst off with X'_w. s is also not envied as only i could envy it, but i receives an edge of greater or equal value to (i,s). This follows since (i,j) is available in $\mathbf{X}$, and by assumption i does not prefer (i,s) to (i,j). Hence, in $\mathbf{X}'$ there exist two non-envied vertices, $k = s$ and $k' = i$, where k receives $X'_k = (k,k')$.

  Property (1) trivially holds because no agent is allocated more than one edge in $\mathbf{X}'$. Property (2) holds for i because it receives its best unallocated edge given that every other vertex has its allocation in $\mathbf{X}'$. Property (2) also holds for any other agent because they are not worst off and the only edge that becomes unallocated in $\mathbf{X}'$ is the (r, j) which neither r (who receives a better edge) nor j (who didn't envy r in $\mathbf{X}$) prefer to what they have in $\mathbf{X}'$. Therefore, the lemma follows.

- **Case 2:** *i prefers (i, s) to (i, j).* We derive $\mathbf{X}'$ as follows:
  - Assign to every vertex $r_\ell \neq i$, for $\ell \in [1, q]$, the edge $(r_\ell, r_{\ell-1})$, and to every vertex $s_\ell \neq j$, for $\ell \in [1, p]$, the edge $(s_\ell, s_{\ell-1})$.
  - i is assigned its most preferred currently unallocated edge adjacent to it; the set of those edges is $U_i(\mathbf{X}) \cup \{(i, s)\}$ since (i, s) was released by vertex s when assigned the new allocation.
  - j is assigned all currently unallocated edges adjacent to it. Note that (i, j) was not the best edge for i, as i prefers the edge (i, s) to (i, j), according to our assumption. Therefore, $U_j(\mathbf{X})$ is available when considering j. Also note that (j, r) was released by vertex r when assigned the new allocation. So, j is assigned the set $X'_j = U_j(\mathbf{X}) \cup \{(j, r)\}$.

  Figure 4(c) describes $\mathbf{X}'$ in this setting.

  We first argue that j is better off. Since r belongs to the safe set of i (Claim 10), r cannot belong to the safe set of j due to our initial assumption and therefore, j prefers X'_j to $X_j = \{(j, s_1)\}$. Using the same arguments as in the case of Figure 3, all vertices that change their allocation but i are better off and all vertices but r_1 are not envied. This means that under $\mathbf{X}'$ there exist two non-envied vertices $k = s_1, k' = j$ where k receives $X'_k = (k, k')$.

  Property (1) trivially holds because the only agent allocated more than one edge under $\mathbf{X}'$ is j who is not envied (using the same arguments we made for i in the analysis of Figure 3). Property (2) holds for i because it receives its best unallocated edge remaining after every other vertex but j has its allocation in $\mathbf{X}'$; then j is given some other unallocated edges which cannot invalidate Propterty (2) for agent i. Property (2) trivially holds for agent j because $U_j(\mathbf{X}') = \emptyset$. Property (2) also holds for all other agents $\neq i, j$ because they are not worst off and the only edge that possibly becomes unallocated in $\mathbf{X}'$ is the (i, s) that neither s (who receives a better edge) nor i (who picks the best unallocated edge) prefer to what they have in $\mathbf{X}'$. Therefore, the Claim follows.

□

We are now ready to complete the proof of Lemma 5 which can be immediately derived by the above lemmata.

Proof of Lemma 5: The allocation $\mathbf{X}$ derived by Algorithm 1 satisfies properties (1) and (2) by Lemma 4. If it doesn't satisfy property (3), we can run Algorithm 2 at least once. The allocation derived after running a single round of Algorithm 2 satisfies the lemma's statement due to Lemma 9. If $\mathbf{X}$ satisfies property (3) but not property (4), the lemma follows due to Lemma 11.

4.2.4 Satisfying all four properties

Here we show how we can derive an allocation that satisfies all four properties from the allocation with the guarantees of Lemma 5.

Lemma 12. *Consider a partial allocation $\mathbf{X}$ satisfying properties (1) and (2) where there exists a non-envied vertex k such that $X_k = \{(k, k')\}$, where k' is also non-envied. Then, after applying Algorithm 2 on $\mathbf{X}$ we get an allocation satisfying all four properties (1)-(4).*

Proof. Properties (1) and (2) hold due to Claim 6. Property (3) holds since the outer while loop of Algorithm 2 will not finish unless property (3) is met. Vertex k was not envied before running Algorithm 2. The only way the allocation to vertex k can change is in the inner while loop. In that case, by Claim 8, k is not envied and receives and edge where the other endpoint is not envied. If k does not change the allocation it remains unenvied and so does k' (by Claim 7). We now argue that, after applying Algorithm 2, vertex k belongs to the safe sets of all envied vertices. This follows from Claim 10, because properties (1)-(3) hold and k receives an edge where the other endpoint is non-envied, so it belongs to the safe set of all envied vertices. Therefore, property (4) is also satisfied. □

4.2.5 Deriving a complete EFX allocation

We finally show that starting from an (partial) allocation satisfying all four properties we can derive a complete EFX allocation (in fact only properties (1), (2) and (4) are needed at this point).

Lemma 13. *If a partial allocation* $\mathbf{X}$ *satisfies all four properties* (1)-(4)*, we can allocate the rest of the unallocated edges and derive an EFX allocation.*

Proof. We obtain a complete EFX allocation by "parking" all unallocated edges somewhere:

- Unallocated edges with at least one non-envied endpoint are assigned to that endpoint, if both endpoints are non-envied then it is assigned arbitrarily to one of the endpoints. This cannot generate envy as every vertex prefers their allocation to any single unallocated edge and no non-envied vertex v has two of its adjacent edges assigned to any single other vertex. The only remaining unallocated edges are between two envied vertices.

- For two envied vertices i and j, the unallocated edge (i, j) can be allocated to any non-envied vertex in $u \in S_i(\mathbf{X}) \cap S_j(\mathbf{X})$. In fact, u may also be in the intersection of (say) $S_i(\mathbf{X})$ and $S_{j'}(\mathbf{X})$, where j' is also envied, and u may also be assigned the edge (i, j'). However, no matter how many adjacent unallocated edges to i are assigned to u, i will not envy u. This follows from the definition of the safe set.

□

Next we summarize how we can derive a complete EFX allocation for any graph:

Proof of Theorem 3: Let $\mathbf{X}$ be the partial orientation computed by the greedy Algorithm 1. By Lemma 4 we have that $\mathbf{X}$ satisfies properties (1) and (2). If $\mathbf{X}$ doesn't also satisfied both properties (3) and (4), we can derive a new allocation $\mathbf{X}'$ such that properties (1) and (2) are still satisfied and there exists a non-envied vertex k that receives a single edge where the other endpoint is also non-envied (Lemma 5). Then from $\mathbf{X}'$ we can derive an allocation $\mathbf{X}''$ satisfying all four properties (Lemma 12). So, overall, either $\mathbf{X}$ or $\mathbf{X}''$ satisfies all four properties, and by Lemma 13 we can derive a complete EFX allocation.

It is easy to verify that all of these algorithms and procedures run in polynomial time. □

5 Discussion and Open Problems

Our model captures the setting where every item is relevant to at most two agents and for every pair of agents there is at most one relevant item in common. There are natural generalizations to one or both of these conditions, where (a) every item is relevant to at most some small number of agents, say, p agents – this can be modeled by considering hyperedges of size p rather than edges – and/or (b) that every pair of agents (or small set of p agents) has an upper bound q on the number of items relevant to both (or all) of them – this can be captured by allowing a q multiplicity of edges (or q multiplicity of size p hyperedges). In our graph setting we have $p = 2$ and $q = 1$ and have shown that EFX allocations do exist in this case. We do not know if EFX allocations exist for these generalizations. In particular, for appropriate parameters this would imply that EFX exists without conditions so — unless one can solve the basic question if EFX allocations always exist — partial progress can possibly be made for larger values of p and q.

Another question of interest is understanding for what classes of graphs an EFX orientation is guaranteed to exist. E.g., an EFX orientation always exists in trees, cycle graphs, and multistars.

A property of our solutions is that items are "parked" at vertices that have no value for them. In fact, one can reinterpret the fact that orientations do not always exist as proof that such inefficiency is inherent (in the full problem as well as in our restriction). Thus, it is obvious to want to consider the efficiency of the allocation. If follows from the hardness result that minimizing the number of parked items is not polytime but what about approximations (for EFX allocations on graphs, or extensions thereof)? What can one say about approximating social welfare? And about Nash social welfare? These problems on graphs may be easier than the same problems in the general setting.

Finally, one hopes that insights for EFX allocations on graphs may help make progress on the general problem of EFX allocations.

References

[Akrami et al.(2022)] Hannaneh Akrami, Noga Alon, Bhaskar Ray Chaudhury, Jugal Garg, Kurt Mehlhorn, and Ruta Mehta. 2022. EFX Allocations: Simplifications and Improvements. *CoRR* abs/2205.07638 (2022).

[Amanatidis et al.(2022)] Georgios Amanatidis, Haris Aziz, Georgios Birmpas, Aris Filos-Ratsikas, Bo Li, Hervé Moulin, Alexandros A. Voudouris, and Xiaowei Wu. 2022. Fair Division of Indivisible Goods: A Survey. *CoRR* abs/2208.08782 (2022).

[Berendsohn et al.(2022)] Benjamin Aram Berendsohn, Simona Boyadzhiyska, and László Kozma. 2022. Fixed-point cycles and EFX allocations. *CoRR* abs/2201.08753 (2022).

[Berger et al.(2022)] Ben Berger, Avi Cohen, Michal Feldman, and Amos Fiat. 2022. Almost Full EFX Exists for Four Agents. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022.* AAAI Press, 4826–4833.

[Budish(2011)] Eric Budish. 2011. The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes. *Journal of Political Economy* 119, 6 (2011), 1061–1103. `https://doi.org/10.1086/664613`

[Caragiannis et al.(2019)] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. 2019. The unreasonable fairness of maximum Nash welfare. *ACM Transactions on Economics and Computation (TEAC)* 7, 3 (2019), 1–32.

[Chaudhury et al.(2020)] Bhaskar Ray Chaudhury, Jugal Garg, and Kurt Mehlhorn. 2020. EFX exists for three agents. In *Proceedings of the 21st ACM Conference on Economics and Computation.* 1–19.

[Chaudhury et al.(2021a)] Bhaskar Ray Chaudhury, Jugal Garg, Kurt Mehlhorn, Ruta Mehta, and Pranabendu Misra. 2021a. Improving EFX Guarantees through Rainbow Cycle Number. In *EC '21: The 22nd ACM Conference on Economics and Computation, Budapest, Hungary, July 18-23, 2021*, Péter Biró, Shuchi Chawla, and Federico Echenique (Eds.). ACM, 310–311. `https://doi.org/10.1145/3465456.3467605`

[Chaudhury et al.(2021b)] Bhaskar Ray Chaudhury, Telikepalli Kavitha, Kurt Mehlhorn, and Alkmini Sgouritsa. 2021b. A Little Charity Guarantees Almost Envy-Freeness. *SIAM J. Comput.* 50, 4 (2021), 1336–1358.

[Chen et al.(2009)] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. 2009. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM (JACM)* 56, 3 (2009), 1–57.

[Christodoulou et al.(2022)] George Christodoulou, Elias Koutsoupias, and Annamária Kovács. 2022. On the Nisan-Ronen conjecture. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS).* IEEE, 839–850.

[Christodoulou et al.(2023)] George Christodoulou, Elias Koutsoupias, and Annamária Kovács. 2023. A proof of the Nisan-Ronen conjecture. *arXiv preprint* (2023).

[Daskalakis et al.(2009)] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. 2009. THE COMPLEXITY OF COMPUTING A NASH EQUILIBRIUM. *SIAM J. Comput.* 39, 1 (2009), 195.

[Jahan et al.(2022)] Shayan Chashm Jahan, Masoud Seddighin, Seyed Mohammad Seyed Javadi, and Mohammad Sharifi. 2022. Rainbow Cycle Number and EFX Allocations: (Almost) Closing the Gap. *CoRR* abs/2212.09482 (2022).

[Lipton et al.(2004)] Richard J. Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. 2004. On Approximately Fair Allocations of Indivisible Goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce* (New York, NY, USA) *(EC '04).* Association for Computing Machinery, New York, NY, USA, 125–131. `https://doi.org/10.1145/988772.988792`

[Mahara(2021)] Ryoga Mahara. 2021. Extension of Additive Valuations to General Valuations on the Existence of EFX. In *ESA (LIPIcs, Vol. 204)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 66:1–66:15.

[Nisan and Ronen(2001)] Noam Nisan and Amir Ronen. 2001. Algorithmic Mechanism Design. *Games and Economic Behavior* 35 (2001), 166–196.

[Payan et al.(2022)] Justin Payan, Rik Sengupta, and Vignesh Viswanathan. 2022. Relaxations of Envy-Freeness Over Graphs. (2022). `https://doi.org/10.48550/ARXIV.2202.10946`

[Plaut and Roughgarden(2020)] Benjamin Plaut and Tim Roughgarden. 2020. Almost envy-freeness with general valuations. *SIAM Journal on Discrete Mathematics* 34, 2 (2020), 1039–1068.

[Procaccia(2020)] Ariel D. Procaccia. 2020. Technical Perspective: An Answer to Fair Division's Most Enigmatic Question. *Commun. ACM* 63, 4 (March 2020), 118. `https://doi.org/10.1145/3382131`