An EFX Allocation Protocol for Restricted Additive Valuations

Hannaneh Akrami1,2*, Rojin Rezvan3 and Masoud Seddighin4
1Max Planck Institute for Informatics
2Graduiertenschule Informatik, Universität des Saarlandes
3 University of Texas at Austin, Computer Science Department
4School of Computer Science, Institute for Research in Fundamental Sciences (IPM), P. O. Box: 19395 - 5746, Tehran, Iran
hakrami@mpi-inf.mpg.de, rojinrezvan@utexas.edu, seddighin@ipm.ir

Abstract

We study the problem of fairly allocating a set of indivisible goods to a set of n agents. Envy-freeness up to any good (EFX) criterion (which requires that no agent prefers the bundle of another agent after the removal of any single good) is known to be a remarkable analogue of envy-freeness when the resource is a set of indivisible goods. In this paper, we investigate EFX for restricted additive valuations, that is, every good has a non-negative value, and every agent is interested in only some of the goods.

We introduce a natural relaxation of EFX called EFX which requires that no agent envies another agent after the removal of any k goods. Our main contribution is an algorithm that finds a complete (i.e., no good is discarded) EFX allocation for restricted additive valuations. In our algorithm we devise new concepts, namely configuration and envy-elimination that might be of independent interest.

We also use our new tools to find an EFX allocation for restricted additive valuations that discards at most \([n/2] - 1\) goods.

1 Introduction

Fair allocation deals with the problem of allocating a resource to agents with diverse preferences. Due to the wide range of applications, this problem has received attention in different fields such as economics, mathematics, and computer science [Lipton et al., 2004; Brams and Taylor, 1996].

In the early studies, the resource was assumed to be a heterogeneous divisible cake. This case has been mostly considered by mathematicians and economists under the title of “Cake Cutting”. In contrast, recent studies often focus on more practical cases where the resource is less divisible. Such instances arise in many real-world scenarios, e.g., dividing the inherited wealth among heirs, divorce settlements, etc [Endriss, 2017; Brandt et al., 2016]. In the indivisible case, the resource is a set \(M\) of indivisible goods that must be divided among \(n\) agents. Each agent \(i\) has a valuation function \(v_i: 2^M \rightarrow \mathbb{R}\) and the goal is to allocate the goods to the agents while satisfying a certain fairness objective.

Envy-freeness is one of the most well-studied fairness notions in the literature. Formally, let \(X = \langle X_1, X_2, \ldots, X_n \rangle\) be an allocation that allocates bundle \(X_i\) to agent \(i\). We say agent \(i\) envies agent \(j\), if \(v_i(X_i) < v_i(X_j)\). An allocation is envy-free, if no agent envies another agent.

Perhaps one of the reasons that envy-freeness is widely accepted among economists is that, despite strict conditions, there are strong guarantees for this notion in the divisible setting. For example, there always exists an envy-free allocation of the cake such that each agent receives a connected piece [Stromquist, 1980]. However, beyond divisibility when dealing with a set of indivisible goods, this notion is too strong to be satisfied. For example, consider an instance with two agents and one good for which both agents have positive valuation. Such barriers have led to natural relaxations of envy-freeness that are more suitable for the case of indivisible goods. Envy-freeness up to one good (EF1) and envy-freeness up to any good (EFX) are among the most prominent relaxations of envy-freeness for indivisible goods. The idea behind these two notions is to allow a limited amount of envy among the agents. Formally, given allocation \(X = \langle X_1, X_2, \ldots, X_n \rangle\), we say \(X\) is

- EFX (Budish [2011]): if for every agents \(i\) and \(j\), there exists a good \(g \in X_j\) s.t. \(v_i(X_j) \geq v_i(X_j \setminus \{g\})\).

- EFX (Caragiannis et al. [2019b]): if for every agents \(i\) and \(j\), and every good \(g \in X_j\), \(v_i(X_j) \geq v_i(X_j \setminus \{g\})\).

By definition, every envy-free allocation is also EFX, and every EFX allocation is also EF1. So far, we know that when the valuation functions are monotone, EF1 allocations always exist and can be found in polynomial time [Lipton et al., 2004]. In sharp contrast, it turns out that the EFX notion is much more challenging. Currently, the existence of an EFX allocation is only proved for very limited cases [Plaut and Roughgarden, 2020; Chaudhury et al., 2020a; Barman et al., 2018b; Amanatidis et al., 2021].

Recent findings on EFX suggest that avoiding a subset of goods may result in strong EFX guarantees. This subject (also known as EFX with charity) is pioneered by the work of Cara-

*Contact Author

1The original definition of EFX assumes that the removed good \(g\) has an additional property that \(v_i(X_i \setminus \{g\}) < v_i(X_j)\). However, our existential results in Sections 5 and 6 work for the more general case where \(v_i(X_i \setminus \{g\}) \leq v_i(X_j)\).
giannis et al. [2019a] wherein the authors show that there exists an allocation that discards a subset of goods and finds an EFX allocation for the rest of the goods such that the Nash welfare\(^2\) of the allocation is at least half of the optimal Nash welfare. Several follow-up works have reduced the number and the total value of the discarded goods [Chaudhury et al., 2020b; Berger et al., 2021].

In this paper, we focus on the EFX notion and its relaxations (including EFX with charity) when the valuation functions are restricted additive. The restricted additive setting is an important subclass of additive setting that has gained popularity in allocation problems during the past decade and particularly for maximin fairness notion [Asadpour et al., 2012; Svensson, 2012; Feige, 2008; Jansen and Rohwedder, 2017; Bansal and Sviridenko, 2006; Cheng and Mao, 2018; Anamalai et al., 2017; Chaudhury and Mao, 2019; Khot and Ponnuswami, 2007]. In the restricted additive setting, the assumption is that the valuation functions are additive, and furthermore, each good \(g\) has an inherent value \(v(g)\) so that for any agent \(i\), we have \(v_i(g) \in \{0, v(g)\}\).

### 1.1 Our Contribution

We start by introducing EF\(k\)X notion which is indeed a relaxation of EFX that allows an amount of envy up to the value of \(k\) least valuable goods of a bundle. Our main result is Algorithm 2 that finds a complete EF2X allocation for restricted additive valuations. The algorithm consists of 4 updating rules plus an additional final step. As long as it is possible, we update the allocation using one of the updating rules. When none of the rules is applicable, we perform the additional step to obtain a complete allocation. The rules are based on new concepts, namely configuration and envy-elimination which we describe in the following.

**Configuration.** One important point of departure of our method from the existing techniques is that alongside updating the allocation, we maintain a partitioning of the agents into several groups. This partitioning has the property that the value of the agents in the same group are close to each other. We use the term configuration to refer to a pair of an allocation and a partition. The updating process at each step takes a configuration as input and updates both the allocation and the partition. Well-established concepts such as champion and champion-graph are also revised in accordance with the definition of configuration.

**Envy-elimination.** At the heart of our updating rules we exploit a process called envy-elimination. Envy-elimination is designed to circumvent deadlocks. At several points during the algorithm we might allocate a good to an agent that violates the EFX property. In such situations, we execute the envy-elimination process. This process restores the EFX property by merely eliminating goods from the bundles of agents. Therefore, at the end of this process, the value of each agent for her bundle is not more than her value beforehand.

**A new Potential Function.** Note that the fact that the social welfare strictly decreases after the envy-elimination process might question the termination of our algorithm. This brings us to another challenge: we must show that the algorithm ends after a finite number of updates. To prove the termination of the algorithm, we introduce a potential function \(\Phi\) which maps a pair \(\sigma = (X, R)\) of a partial allocation \(X\) and a partition \(R\) of agents to a vector \(\Phi(\sigma)\) of rational numbers and show that after each update, \(\Phi(\sigma)\) increases lexicographically. This indicates that the updating process terminates after a finite number of updates. Note that for a given configuration \(\sigma\), function \(\Phi(\sigma)\) relies on both \(X\) and \(R\).

We later turn our attention beyond EF2X to see whether our new tools can be used to obtain better guarantees for the EFX notion. Our second result is Algorithm 3 that finds a partial EFX allocation which discards less than \([n/2]\) goods. The currently best known result in this direction is the work of Berget et al. [2021] which proves the existence of a partial EFX allocation that discards at most \(n - 2\) goods when the valuation of agents is restricted to some superclass of additive valuations. This result was further improved to general valuations by Mahara [2021]. Compared to them, our algorithm reduces the number of discarded goods for the restricted additive setting by a factor of 2.

### 2 Related Work

We refer the reader to [Brams and Taylor, 1996] for an overview on fair division, classic fairness notions, and related results. The notion we study in this paper is EFX which is a relaxation of envy-freeness for the case that the resource is a set of indivisible goods. EFX is among the most studied fairness notions in recent years and “Arguably, the best fairness analog of envy-freeness for indivisible items” [Caragiannis et al., 2019b]. This notion originates in the work of Caragiannis et al. wherein the authors provide some initial results on EFX and its relation to other notions. However, despite extensive investigations, the existence of a complete EFX allocation is only proved for very limited cases: when the number of agents is 2 or 3 [Plaut and Roughgarden, 2020; Chaudhury et al., 2020a], and when the valuations are either identical [Plaut and Roughgarden, 2020], binary [Barman et al., 2018b], or bi-valued [Amanatidis et al., 2021]. Given this impenetrability of EFX, a growing strand of research started considering its relaxations. These relaxations can be classified into three categories:

**Approximation.** One approach is to find allocations that are approximately EFX. The first result in this direction is a 1/2-EFX allocation [Plaut and Roughgarden, 2020] which is later improved to 0.618 [Amanatidis et al., 2020].

**Weakening the fairness requirement.** Recall that in an EFX allocation, any possible envy is removed by eliminating the least valuable good. Recently, Farhadi et al. [2021] suggest a relaxed version of EFX called EFR in which instead of eliminating the least valuable good to evaluate fairness, it eliminates a good uniformly at random. They show that a 0.73-EFR allocation always exists. Another example is envy-freeness up to one less preferred good (EFL) [Barman et al., 2018a] which limits the value of the eliminated good. The EF\(k\)X notion we introduced in Section 1.1 also falls within this category.
Discarding a subset of goods. Another approach is to relax the assumption that the final allocation must allocate all the goods. This line is initiated by the work of Caragiannis et al. [2019a] wherein the authors prove the existence of a partial EFX allocation whose Nash welfare is at least half of the optimal Nash welfare. Following this work, Chaudhury et al. [2020b] proved the existence of a partial EFX allocation that leaves at most \( n - 1 \) goods unallocated. Berger et al. [2021] decreased the number of discarded goods to \( n - 2 \) for a super class of additive valuations and Mahara [2021] extended this result to general valuations. Recently, Chaudhury et al. [2021] presented a framework to obtain a \((1 - \epsilon)\)-EFX allocation with sub-linear number of unallocated goods for \( \epsilon \in (0, \frac{1}{2}) \). Our result in Section 6 falls within this category.

3 Preliminaries

We denote the set of agents by \( N = \{1, 2, \ldots, n\} \) and the set of goods by \( M \). Each agent \( i \) has a valuation function \( v_i : \mathbb{N}^M \to \mathbb{R}^+ \) which represents the value of that agent for each subset of the goods. An allocation is a specification of how goods in \( M \) are divided among the agents. We denote an allocation by \( X = (X_1, X_2, \ldots, X_n) \), where \( X_i \) is the bundle allocated to agent \( i \). Allocation \( X \) is complete if \( \bigcup_i X_i = M \) and is partial otherwise. For a partial allocation \( X \), we refer to the set of goods that are not allocated to any agent as the pool of unallocated goods and denote it by \( P_X \). When \( X \) is clear from the context, we simply use \( P \) instead of \( P_X \). We say a good \( g \) is wasted, if it is allocated to an agent that has zero value for \( g \). Typically, non-wasteful allocations refer to allocations that admit no wasted good.

In this paper, we are interested in allocations that satisfy certain fairness properties. In Section 1, we defined the EFX notion. Here we define a more general form of EFX, namely, envy-freeness up to any \( k \) goods or EfkX.

Definition 3.1. An allocation \( X = (X_1, \ldots, X_n) \) is EfkX, if for all \( i \neq j \) and every collection of \( \ell = \min(k, |X_j|) \) distinct goods \( g_1, g_2, \ldots, g_\ell \in X_j \) we have \( v_i(X_j \{ g \}) \geq v_i(X_j \setminus \{g_1, g_2, \ldots, g_\ell\}) \).

In particular, an EFX allocation is EfkX with \( k = 1 \). Our main results are related to the cases where \( k = 1 \) and \( k = 2 \).

Restricted Additive Valuations. We consider a special case of valuation functions in which each good \( g \) has an inherent value \( v(g) \) and the contribution of \( g \) to any set of goods for any agent is either 0 or \( v(g) \). Under this restriction, when the valuations are also additive, we call them restricted additive.

Definition 3.2. A set \( \{v_1, v_2, \ldots, v_n\} \) of valuation functions is restricted additive, if for every \( 1 \leq i \leq n, v_i \) is additive, and for every good \( g \in M \), we have \( v_i(g) \in \{0, v(g)\} \).

For a set \( S \) of goods we define \( v(S) \) as \( \sum_{g \in S} v(g) \).

3.1 Configurations

In this work, we represent the status of our algorithm at each step via two ingredients: a partial allocation \( X \) and a partition \( R \) of the agents. A partition of a set is a grouping of its elements into non-empty subsets such that every element is included in exactly one subset. For a partition \( R \), we denote the \( i \)th group of \( R \) by \( R_i \).

For brevity, we use the term configuration to refer to a pair of an allocation and a partition. At each step during our algorithms we take a configuration \((X, R)\) as input and return another configuration \((X', R')\). Our algorithms are developed in such a way that the configurations satisfy three structural properties. Property 1 requires that the allocation is EFX.

Property 1 (EFX). A configuration \( \sigma = (X, R) \) is EFX, if allocation \( X \) is EFX.

Note that Property 1 is independent of the partition. The second property we consider for configurations incorporates both the allocation and the configuration.

Property 2 (Envy-compatibility). A configuration \((X, R)\) is envy-compatible, if for all \( 1 \leq \ell \leq |R| \) we have

\[
\forall i \in \ell_i \cap g \in X_i, v_i(X_i \setminus \{g\}) \leq v_{\ell_i}(X_{\ell_i}),
\]

where

\[
r_{\ell_i} = \arg\min_{j \in \ell_i} v_j(X_j).
\]

For an envy-compatible configuration \((X, R)\), for every \( 1 \leq \ell \leq |R| \) we define \( r_{\ell_i} \) as in Equation (1) and call \( r_{\ell_i} \) the representative of group \( R_i \). Furthermore, we suppose that the groups in \( R \) are sorted according to the utility of their representatives, i.e., \( v_{r_1}(X_{r_1}) \leq v_{r_2}(X_{r_2}) \leq \ldots \leq v_{r_{|R|}}(X_{r_{|R|}}) \).

Note that for an allocation \( X \), there might be several different partitions \( R \) such that configuration \((X, R)\) is envy-compatible. For example, one trivial such partition is to put each agent into a separate group. However, our interest is the configurations that are more specific. In addition to Properties 1 and 2 our desired configurations admit another important property. Before introducing this property, we must define the notions of champion and champion-graph. These concepts were first introduced in [Chaudhury et al., 2020a]. We revise their definition to incorporate configurations.

Definition 3.3. Let \((X, R)\) be a configuration satisfying Property 2 and let \( i \in R_{\ell_i} \) be an agent. Then, for every subset \( S \) such that \( v_i(S) > v_{r_i}(X_{r_i}) \), we define \( S | i \) to be a smallest subset of \( S \) such that \( v_i(S | i) > v_{r_i}(X_{r_i}) \). In case of multiple options for \( S | i \), we pick one arbitrarily.

We now define champion and champion-graph as follows.

Definition 3.4 (Champion and Champion-graph). Given a configuration \( \sigma = (X, R) \) satisfying Property 2, we say \( i \in R_{\ell_i} \) is a champion of \( S \) if \( v_i(S) > v_{r_i}(X_{r_i}) \) and for every agent \( j \neq i \) with \( j \in R_k \) and \( g \in [S | i] \) we have \( v_{r_k}(X_{r_k}) \geq v_j([S | i] \setminus \{g\}) \). We also define the champion-graph of \( \sigma \), denoted by \( H_{\sigma} \) as follows: for every agent \( i \) there is a vertex in \( H_{\sigma} \). Edges of \( H_{\sigma} \) are of two types:

- Regular edges: for every pair of agent \( i, j \) with \( i \in R_{\ell_i} \) there is a directed edge from \( i \) to \( j \), if \( v_i(X_j) > v_{r_j}(X_{r_j}) \).
- Champion edges: for every pair of agents \( i, j \) and every unallocated good \( g \), there is a directed edge from \( i \) to \( j \) with label \( g \), if \( i \) is a champion of \( X_j \cup \{g\} \).

The last property we consider for configurations is based on the champion-graph.
Lemma 4.1. Before the process. However, as we show in Lemma 4.1, the result of running envy-elimination process on $X$ satisfies Properties 1, 2 and 3.

Note that, by the way we construct $X$, the process first retakes all the wasted goods. Then, the process takes an allocation $X'$ of set $X$. Let $X$ be a configuration satisfying Properties 2 and 3.

Property 3 (Admissibility). A configuration $\sigma = (X, R)$ satisfying Property 2 is admissible if the following hold:

- For all $1 \leq \ell \leq |R|$ and $i \in R_{\ell}$, there is a path from $r_{\ell}$ to $i$ in $H_\sigma$ using only regular edges.
- For all $\ell < k$, $i \in R_{\ell}$ and $j \in R_{k}$, $v_{r_{\ell}}(X_{r_{\ell}}) \geq v_{i}(X_{i})$ holds. I.e., there is no regular edge from $r_{\ell}$ to $r_{k}$.

Lemma 3.5. Let $X$ be a non-wasteful allocation, and let $(X, R)$ be a configuration satisfying Properties 2 and 3. Then, $X$ is EF\text{X}.

### 4 Envy-elimination

One important part of our algorithm is an auxiliary process which we call envy-elimination. This process is designed to bypass possible deadlocks in the updating processes. This process takes an allocation $X$ as input and returns a configuration $(X', R')$ satisfying Properties 1, 2 and 3. To do so, the process first retakes all the wasted goods. Then, the process chooses an agent with the minimum valuation, i.e., \arg \min_{i \in H_\sigma} v_i(X_i). This agent is selected as the representative of set $R_1$, such that allocation $X'$ with $X' = X \cup \{s\}$ for all $s \in P$ is EF\text{X}. Algorithm 2 shows a pseudocode of our algorithm. In the rest of this section, we first describe the updating rules. For each rule, we prove that the resulting configuration satisfies Properties 1, 2, and 3. Also, we prove that all the rules are \Phi\text{-improving.} Finally, we describe how we can allocate the remaining goods to maintain the EF\text{X} property.

#### 5 An EF\text{X} Allocation Algorithm

In this section, we present our algorithm for finding a complete EF\text{X} allocation. Our algorithm consists of four updating rules $U_0$, $U_1$, $U_2$, and $U_3$. We start with configuration $\sigma = (X, R)$, where $X$ is an empty allocation and $R = \langle \{1\}, \{2\}, \ldots, \{n\} \rangle$. At each step, we take the current configuration $\sigma = (X, R)$ as input and update it using one of these rules. These rules are designed in a way that they always (lexicographically) increase the value of $\Phi(\sigma)$, where

$$\Phi(\sigma) = \left[ v_{r_1}(X_{r_1}), v_{r_2}(X_{r_2}), \ldots, v_{r_{|R|}}(X_{r_{|R|}}) \right] + \infty,$$

$$\sum_{i \notin \{r_1, r_2, \ldots, r_{|R|}\}} |X_i| \cdot \sum_{i \in \{r_1, r_2, \ldots, r_{|R|}\}} |X_i|.$$

Therefore, after a finite number of updates we obtain a configuration to which none of the updating rules is applicable. Note that, since we guarantee that $\sigma$ is envy-compatible, $\Phi(\sigma)$ is well-defined. Recall that $r_1, r_2, \ldots, r_{|R|}$ are the representatives of partition $R$.

Afterwards, we show that the remaining goods have a special property that allows us to allocate them to the agents without violating the EF\text{X} property. In other words, if for an allocation $X$ during the algorithm none of the rules is applicable and goods $g_1, \ldots, g_{P'}$ are not allocated, then we can find $|P|$ different agents $i_1, i_2, \ldots, i_{|P|}$ such that allocation $X'$ with $X' = X \cup \{s\}$ for all $s \notin \{i_1, i_2, \ldots, i_{|P|}\}$ is EF\text{X}. Algorithm 2 shows a pseudocode of our algorithm.

In the rest of this section, we first describe the updating rules. For each rule, we prove that the resulting configuration satisfies Properties 1, 2, and 3. Also, we prove that all the rules are $\Phi$-improving. Finally, we describe how we can allocate the remaining goods to maintain the EF\text{X} property.

#### 5.1 Rule $U_0$

We start with rule $U_0$ which is our most basic rule. This rule allocates an unallocated good $g$ to a representative, such that the resulting configuration satisfies Properties 1, 2 and 3. In the following, you can find a schematic view of Rule $U_0$.

---

**Preconditions:**

- $\sigma = (X, R)$ satisfies Properties 1, 2 and 3.
- There exists a good $g \in P$ and an index $\ell$ s.t. $v_{r_\ell}(g) = 0$ and for every $1 \leq k \leq |R|$ and $i \in R_k$, $v_i(X_{r_k} \cup \{g\}) \leq v_{r_\ell}(X_{r_\ell})$.

**Process:**

- Allocate $g$ to $r_\ell$.
- Set $R' = R$.

**Guarantees:**

- $\sigma' = (X', R')$ satisfies Properties 1, 2, and 3.
- $\Phi(\sigma') \geq \Phi(\sigma)$.
Algorithm 2 Complete EF2X allocation

Input: instance \((N, M, (v_1, \ldots, v_n))\)
Output: allocation \(X\)

1. \(X \leftarrow \{\emptyset, \emptyset, \ldots, \emptyset\}\)
2. \(R \leftarrow \{\{1\}, \{2\}, \ldots, \{n\}\}\)
3. \textbf{while} \(U_0\) or \(U_1\) or \(U_2\) or \(U_3\) is applicable \textbf{do}
4. \hspace{1em} Let \(i\) be the minimum index s.t. \(U_i\) is applicable
5. \hspace{1em} Update \((X, R)\) via Rule \(U_i\)
6. \hspace{1em} for \(t \leftarrow |R|\) to 1 \textbf{do}
7. \hspace{2em} while \(\exists i \in R_t \cap N\) s.t. \(v_i(P) > 0\) \textbf{do}
8. \hspace{3em} Choose \(i\) with maximum distance from \(R_t\)
9. \hspace{3em} \(g \leftarrow \arg \max_{h \in P} v_i(h)\)
10. \hspace{3em} \(X_i \leftarrow X_i \cup \{g\}\)
11. \hspace{1em} \(N \leftarrow N \setminus \{i\}\)
12. \hspace{1em} Let \(P_X = \{g_1, g_2, \ldots, g_{|P_X|}\}\)
13. \hspace{1em} Let \(i_1, i_2, \ldots, i_{|P_X|}\) be \(|P_X|\) different agents in \(N\)
14. \hspace{1em} for \(\ell \leftarrow 1\) to \(|P_X|\) \textbf{do}
15. \hspace{2em} \(X_{i_{\ell}} \leftarrow X_{i_{\ell}} \cup \{g_{\ell}\}\)
16. \hspace{1em} Return \(X\)

Lemma 5.1. Rule \(U_0\) is \(\Phi\)-improving and preserves Properties 1, 2, and 3.

5.2 Rule \(U_1\)

This updating rule updates the allocation using a special type of cycle in the champion-graph called champion cycle.

Definition 5.2. For a configuration \(\sigma = (X, R)\), a cycle \(C\) in \(H_\sigma\) is a champion cycle, if for every champion edge \(i \xrightarrow{\star} j\) in \(C\), \(j \in \{r_1, r_2, \ldots, r_{|R|}\}\). Moreover, for every two champion edges of \(C\) with labels \(g\) and \(g'\), we have that \(g \neq g'\).

In \(U_1\), we find a champion cycle in \(H_\sigma\) and update the allocation through this cycle. Then, we apply envy-elimination.

Preconditions:

- \(\sigma = (X, R)\) satisfies Properties 1, 2 and 3.
- There exists a champion cycle \(C\) in \(H_\sigma\).

Process:

- For every edge \(i \rightarrow j \in C\), set \(\tilde{X}_i = X_j\).
- For every champion edge \(i \xrightarrow{\star} j \in C\), set \(\tilde{X}_i = [X_j \cup \{g\} \mid i]\).
- For every agent \(j \notin C\), let \(\tilde{X}_j = X_j\).
- Apply envy-elimination on \(\tilde{X}\).

Guarantees:

- \(\sigma' = (X', R')\) satisfies Properties 1, 2, and 3.
- \(\Phi(\sigma') \geq \Phi(\sigma)\).

Since we apply envy-elimination in the last step of Rule \(U_1\), Lemma 4.1 implies that the resulting configuration satisfies Properties 1, 2, and 3.

Lemma 5.3. Rule \(U_1\) is \(\Phi\)-improving and preserves Properties 1, 2, and 3.

5.3 Rule \(U_2\)

In this updating rule, we check if there exists a representative such that her value for at least one of the remaining goods is non-zero. If so, we simply allocate it to that representative and apply envy-elimination.

Preconditions:

- \(\sigma = (X, R)\) satisfies Properties 1, 2 and 3.
- There exists a representative \(r_\ell\) and an unallocated good \(g\) s.t. \(v_{r_\ell}(g) = v(g)\).
- Rules \(U_0\) and \(U_1\) are not applicable.

Process:

- Allocate \(g\) to \(r_\ell\).
- Apply envy-elimination.

Guarantees:

- \(\sigma' = (X', R')\) satisfies Properties 1, 2, and 3.
- \(\Phi(\sigma') \geq \Phi(\sigma)\).

Lemma 5.4. Rule \(U_2\) is \(\Phi\)-improving and preserves Properties 1, 2, and 3.

5.4 Rule \(U_3\)

Unlike the previous rules, in Rule \(U_3\) the agent to whom we allocate a good is not a representative in \(\sigma\). Here we find an unallocated good \(g\) and a non-representative agent \(i\) such that \(v_i(g) = v(g)\) and allocating \(g\) to agent \(i\) violates the EF2X property.

Preconditions:

- \(\sigma = (X, R)\) satisfies Properties 1, 2 and 3.
- Rules \(U_0\), \(U_1\), and \(U_2\) are not applicable.
- There exists \(i \in R_\ell \setminus \{r_\ell\}\) with distance \(d\) from \(r_\ell\) in \(H_\sigma\) and another \(i' \in R_{\ell'}\) with distance \(d'\) from \(r_{\ell'}\) in \(H_\sigma\) such that \((\ell, d) \geq (\ell', d')\) and there exists \(g \in P\) such that \(v_i(g) = v(g)\) and \(i'\) envies \(X_i \cup \{g\}\) even after removal of 2 goods.

Process:

- Allocate \(g\) to \(i\).
- Apply envy-elimination on \(X\).

Guarantees:

- \(\sigma' = (X', R')\) satisfies Properties 1, 2, and 3.
- \(\Phi(\sigma') \geq \Phi(\sigma)\).

Lemma 5.5. Rule \(U_3\) is \(\Phi\)-improving and preserves Properties 1, 2, and 3.
In this section we present an algorithm that leaves aside at most \(\lfloor n/2 \rfloor - 1\) many goods and finds an EFX allocation for the rest of the goods. Algorithm 3 shows a pseudocode of our allocation method. Similar to Algorithm 2, Algorithm 3 starts with empty allocation \(X\) and partition \(R = \{\{1\}, \{2\}, \ldots, \{n\}\}\). Our algorithm consists of four updating rules \(U_0, U_1, U_2\) and \(U_4\). At each step, we take the current configuration as input and update it using one of these updating rules. Rule \(U_0, U_1,\) and \(U_2\) are already defined in Section 5. Similar to these rules, Rule \(U_4\) is designed in a way that the value of \(\Phi(\sigma)\) increases after each update. When none of these rules is applicable, the algorithm terminates.

In the rest of this section, we describe the new updating rule, namely \(U_4\) and its properties. Next, based on these properties we prove Theorem 6.3 which states that the final allocation is EFX and discards less than \(\lfloor n/2 \rfloor\) goods.

\[\text{Algorithm 3 EFX allocation with } \leq \lfloor n/2 \rfloor \text{ discarded goods}\]

Input: instance \((N, M, (v_1, \ldots, v_n))\)
Output: allocation \(X\)

1. \(X \leftarrow \emptyset, \emptyset, \ldots, \emptyset\)
2. \(R \leftarrow \{\{1\}, \{2\}, \ldots, \{n\}\}\)
3. \textbf{while} \(U_0\) or \(U_1\) or \(U_2\) or \(U_4\) is applicable \textbf{do}
   4. Let \(i\) be the minimum index s.t. \(U_i\) is applicable
   5. Update \((X, R)\) via Rule \(U_i\)
4. Return \(X\)

5.5 Allocating the Remaining Goods
As we mentioned, our algorithm continues to update the allocation as long as one of the rules is applicable. Since there are finitely many possible allocations and the potential function \(\Phi\) increases after each update, we eventually end up with a configuration \((X, R)\) s.t. none of the rules can be applied on \(\sigma\). At that moment, if allocation \(X\) allocates all the goods, we are done. Otherwise, there are some goods remaining in the pool. In this section, we show how to allocate these goods so that the final allocation preserves the EFX property.

Generally, our strategy is to find \(|P|\) different agents and allocate one good to each one of them. The process of allocating the remaining goods is illustrated in Algorithm 2. In this algorithm, we start by \(\ell = |R|\) and as long as there is an agent \(i \in R_t \cap N\) such that \(v_i(P) > 0\), we give \(i\) her most desirable good from the pool\(^4\) and remove \(i\) from \(N\). Then we decrease \(\ell\) by one and repeat the same process. The process goes on until either no good remains in the pool, or \(\ell = 0\). Let us call the set of agents that receive one good in this process by \(N_1\). Let us denote the set of unallocated goods at this stage by \(M_2\). In case that \(\ell = 0\) and the pool is still not empty (i.e., \(M_2 \neq \emptyset\)), we allocate each remaining good to an arbitrary agent in \(N \setminus N_1\) (one good to each agent). We denote the agents that received some good in \(M_2\) by \(N_2\).

Theorem 5.6. Given restricted additive valuations, Algorithm 2 returns a complete EFX2 allocation.

6 An EFX Allocation with at most \([n/2] - 1\) Discarded Goods
In this section we present an algorithm that leaves aside at most \([n/2] - 1\) many goods and finds an EFX allocation for the rest of the goods. Algorithm 3 shows a pseudocode of our allocation method. Similar to Algorithm 2, Algorithm 3 starts with empty allocation \(X\) and partition \(R = \{\{1\}, \{2\}, \ldots, \{n\}\}\). Our algorithm consists of four updating rules \(U_0, U_1, U_2\) and \(U_4\). At each step, we take the current configuration as input and update it using one of these updating rules. Rule \(U_0, U_1,\) and \(U_2\) are already defined in Section 5. Similar to these rules, Rule \(U_4\) is designed in a way that the value of \(\Phi(\sigma)\) increases after each update. When none of these rules is applicable, the algorithm terminates.

In the rest of this section, we describe the new updating rule, namely \(U_4\) and its properties. Next, based on these properties we prove Theorem 6.3 which states that the final allocation is EFX and discards less than \([n/2]\) goods.

4If there were multiple such goods, we select one arbitrarily.

6.1 Rule \(U_4\)
In this rule, we consider the whole set of the goods in the pool to update the input \(\sigma = (X, R)\). This rule states that if the set of the goods in the pool has a champion \(i \in R_t\), we update the allocation as follows: we find a path from \(r_t\) to \(i\) and update the allocation through this path. Then we apply envy-elimination to obtain \(\sigma' = (X', R')\).

Preconditions:
- \(\sigma = (X, R)\) satisfies Properties 1, 2 and 3.
- There exists an agent who envies \(P_X\).

Process: Let \(i \in R_t\) be a champion of \(P\) and let \(r_t = j_1 \rightarrow j_2 \cdots \rightarrow j_p = i\) be a path in \(H_{\sigma}\).
- For all \(1 \leq t \leq p - 1\), set \(X_{j_t} = X_{j_{t+1}}\).
- Set \(\overline{X}_i = |P \setminus i|\).
- Apply envy-elimination on \(\overline{X}\).

Guarantees:
- \(\sigma' = (X', R')\) satisfies Properties 1, 2, and 3.
- \(\Phi(\sigma') \geq \Phi(\sigma)\).

Lemma 6.1. Rule \(U_4\) is \(\Phi\)-improving and preserves Properties 1, 2, and 3.

6.2 Bounding the Number of Discarded Goods
As we mentioned, when none of the updating rules is applicable, we terminate the algorithm. As we proved in Lemmas 5.1, 5.3, 5.4 and 6.1, the allocation after each update is EFX. Therefore, the final allocation preserves the EFX property as well. Also, since the value of \(\Phi(\sigma)\) increases after each update, Algorithm 3 terminates after a finite number of updates. It only remains to prove that the number of remaining goods in the pool is less than \([n/2]\), which we prove in Lemma 6.2.

Lemma 6.2. Let \(P\) be the pool of unallocated goods at the end of Algorithm 3. Then, we have \(|P| < [n/2]\). Also, no agent envies \(P\).

Finally, Lemma 6.2 together with the fact that the final allocation is EFX yields Theorem 6.3.

Theorem 6.3. Assuming that the valuations are restricted additive, Algorithm 3 returns an allocation \(X\) and a pool \(P\) of unallocated goods such that
- \(X\) is EFX, and
- \(v_i(X_i) \geq v_i(P)\) for every agent \(i\), and
- \(|P| < [n/2]\).

References


