Approximately EFX Allocations for Indivisible Chores

Shengwei Zhou , Xiaowei Wu
IOTSC, University of Macau
{yc17423, xiaoweiwu}@um.edu.mo,

Abstract

In this paper we study how to fairly allocate a set of m indivisible chores to a group of n agents, each of which has a general additive cost function on the items. Since envy-free (EF) allocation is not guaranteed to exist, we consider the notion of envy-freeness up to any item (EFX). In contrast to the fruitful results regarding the (approximation of) EFX allocations for goods, very little is known for the allocation of chores. Prior to our work, for the allocation of chores, it is known that EFX allocations always exist for two agents, or general number of agents with identical ordering cost functions. For general instances, no non-trivial approximation result regarding EFX allocation is known. In this paper we make some progress in this direction by showing that for three agents we can always compute a 5-approximation of EFX allocation in polynomial time. For $n \geq 4$ agents, our algorithm always computes an allocation that achieves an approximation ratio of $3n^2$ regarding EFX. We also study the bi-valued instances, in which agents have at most two cost values on the chores, and provide polynomial time algorithms for the computation of EFX allocation when $n = 3$, and $(n-1)$-approximation of EFX allocation when $n \geq 4$.

1 Introduction

Fairness is receiving increasing attention in a broad range of research fields, including but not limited to computer science, economics and mathematics. A fair allocation problem focuses on allocating a set M of m items to a group N of n agents, where different agents may have different valuation functions on the items. When the valuation functions give positive values, the items are considered as goods, e.g., resources; when the valuation functions give negative values, the items are considered as chores, e.g., tasks. In the latter case we refer to the valuation functions as cost functions. In this paper, we consider the functions are additive. Arguably, two of the most well-studied fairness notions are *envy-freeness* (EF) [Foley, 1967] and *proportionality* (PROP) [Steinhaus, 1948]. Proportionality means that each agent received at least her proportional share of all items. Envy-freeness is even stronger. Informally speaking, an allocation is EF if no agent wants to exchange her bundle of items with another agent in order to increase her utility. In contrast to the case of divisible items, where EF and PROP allocations always exist [Aziz and Mackenzie, 2016; Edward Su, 1999; Alon, 1987], when items are indivisible, they are not guaranteed to exist even for some simple cases. For example, consider allocating a single indivisible item to two agents. This example also defies approximations of EF and PROP allocation. Therefore, researchers have turned their attention to relaxations of these fairness notions. *Envy-free up to one item* (EF1) [Lipton *et al.*, 2004] and *envy-free up to any item* (EFX) [Caragiannis *et al.*, 2019] are two widely studied relaxations of EF. Informally speaking, an EF1 allocation requires that the envy between any two agents can be eliminated by removing some item; while an EFX allocation requires that the envy can be eliminated by removing any item.

It has been shown that EF1 allocations are guaranteed to exist and can be found in polynomial time for goods [Lipton *et al.*, 2004], chores and even mixture of the two [Bhaskar *et al.*, 2021]. However, EF1 could sometimes lead to extreme unfairness, even if a much fairer allocation exists. EFX, on the other hand, puts a much stronger constraint on the allocation and is arguably the most compelling fairness notion. There are fruitful results regarding the existence and computation of (approximations of) EFX allocations since the notion was first proposed by Caragiannis et al. [Caragiannis *et al.*, 2019]. For the allocation of goods, it has been shown that EFX allocations exist for two agents with general valuations and any number of agents with identical ordering (IDO) valuations [Plaut and Roughgarden, 2020], and recently for three agents [Chaudhury *et al.*, 2020a]. It remains a fascinating open problem whether EFX allocations always exist in general. For general number of agents with additive valuations, there are efficient algorithms for the computation of 0.5-approximate[1] EFX allocations [Plaut and Roughgarden, 2020; Chan *et al.*, 2019] and 0.618-approximate EFX allocations [Amanatidis *et al.*, 2020].

In contrast to the allocation of goods, very little regarding EFX allocations for chores is known to this day. It can be easily shown that the divide-and-choose algorithm computes an

[1]Regarding approximations of EFX allocations, the approximate ratios are at most 1 for goods, and at least 1 for chores.

EFX allocation when there are two agents with general valuations. Very recently, it is shown that EFX allocations always exist for IDO instances [Li *et al.*, 2022] and levelled preferences [Gafni *et al.*, 2021]. However, even for three agents with general additive valuations, it is unknown whether constant approximations of EFX allocations exist, let alone the existence of EFX allocations.

1.1 Main Results

In this paper, we propose polynomial-time algorithms for the computation of approximately EFX allocations for indivisible chores. For three agents, our algorithm achieves an approximation ratio of 5 while for $n \geq 4$ agents the approximation ratio is $3n^2$. Prior to our work, no non-trivial results regarding the approximation of EFX allocation for chores are known, except for some special cases [Li *et al.*, 2022].

Result 1 (Theorem 3)**.** *There exists a polynomial time algorithm that computes a* 5*-approximate EFX allocation for three agents with additive cost functions.*

Result 2 (Theorem 8)**.** *There exists a polynomial time algorithm that computes a* $3n^2$*-approximate EFX allocation for* $n \geq 4$ *agents with additive cost functions.*

Main Challenge. While being two seemingly similar problems, the EFX allocations of goods and chores admit distinct difficulties in approximability. For the allocation of goods, while computing an EFX allocation seems difficult, getting a constant approximation ratio turns out to be quite straightforward. Existing algorithms [Amanatidis *et al.*, 2020; Plaut and Roughgarden, 2020] for the computation of approximately EFX allocation for goods follow the common framework of *Sequential Picking* equipped with the *Envy-Cycle Elimination* technique [Lipton *et al.*, 2004; Bhaskar *et al.*, 2021]. Roughly speaking, in these algorithms, in each round an "un-envied" agent will be chosen to pick her favourite unallocated item, where the envy-cycle elimination technique ensures that there will always be an un-envied agent in each round. The result of [Li *et al.*, 2022] also follows this framework to compute an EFX allocation for chores on identical ordering instances. The key to the analysis is by showing that the value/cost of an agent increases by at most a small constant factor in each round. However, it seems quite challenging to extend this framework to handle general instances for the allocation of chores. For the allocation of goods, it can be shown that the utilities of agents are non-decreasing in each round: the picking operation and envy-cycle elimination do not decrease the value of any agent. In contrast, for the allocation of chores, when an agent picks an item, its cost increases; when an envy-cycle is eliminated, the cost of the involved agents decreases. This introduces a main difficulty in computing an (approximation of) EFX allocation for chores: if the cost of an agent is very small when it picks an item with large cost, the approximation ratio of EFX can be arbitrarily bad.

Our Techniques. To get around this difficulty, we adopt a completely different approach in dividing the items. Our first observation is that when all items have small costs to all agents, then there is likely to exist a partition that looks "even" to all agents. To handle this case, we propose the Sequential Placement algorithm, which computes a partition of the items for a group of agents such that the ratios between the cost of any two bundles are bounded, under every agent's cost function. On the other hand, if there exists an item e that has large cost to an agent i, then by allocating the item to some other agent j, we can ensure that agent i does not seriously envy agent j, no matter what items agent i received eventually. Our algorithms rely on a careful combination of the above two observations. For three agents, we show that by classifying the agents into different types depending on the number of large items they have, we are able to get a 5-approximate EFX allocation. To extend the ideas to general number of agents, we borrow some existing techniques for the computation of PROP1 allocation for goods [Lipton *et al.*, 2004], and bound the approximation ratio by $3n^2$.

We also show that our results can be improved for *bi-valued* instances. An instance is called a bi-valued instance if there exists two values $a, b \geq 0$ such that $c_i(e) \in \{a, b\}$ for every agent $i \in N$ and item $e \in M$. In other words, each agent classifies the set of items into *heavy* items and *light* items, and items in the same category have the same cost to the agent. The bi-valued instances are commonly considered as one of most important special case of the fair allocation problem [Aziz and Brown, 2020; Akrami *et al.*, 2021; Garg and Murhekar, 2021; Amanatidis *et al.*, 2021; Garg *et al.*, 2021; Ebadian *et al.*, 2021]. For the case of bi-valued goods, polynomial time algorithms have been proposed for the computation of EFX allocations [Amanatidis *et al.*, 2021] and EFX allocations that are also Pareto optimal (PO) [Garg and Murhekar, 2021]. Very recently, it has been shown that for bi-valued chores, EF1 allocations that are PO can be computed in polynomial time [Garg *et al.*, 2021; Ebadian *et al.*, 2021]. Unfortunately, no non-trivial results regarding the approximation of EFX allocations is known for bi-valued chores.

In this paper, we make progress towards answering this problem for bi-valued instances. For three agents, we propose a polynomial time algorithm that always computes an EFX allocation; for $n \geq 4$ agents, we propose a polynomial time algorithm that always computes a $(n-1)$-approximate EFX allocation.

Result 3 (Theorem 15)**.** *There exists a polynomial time algorithm that computes an EFX allocation for three agents with bi-valued cost functions.*

Result 4 (Theorem 19)**.** *There exists a polynomial time algorithm that computes a* $(n-1)$*-approximate EFX allocation for* $n \geq 4$ *agents with bi-valued cost functions.*

1.2 Other Related Works

For the allocation of goods, there are also works that study partial EFX allocations, i.e., EFX allocations with some of the items unallocated. To name a few, Chaudhury et al. [Chaudhury *et al.*, 2020b] show that EFX allocations exist if we are allowed to leave at most $n-1$ items unallocated. The result has recently been improved to $n-2$ items by Berger et al. [Berger *et al.*, 2021]. They also show that an EFX allocation that leaves at most one item unallocated exists for four

agents. Very recently, Chaudhury et al. [Chaudhury *et al.*, 2021] show that a $(1-\epsilon)$-approximate EFX allocation with sublinear number of unallocated goods and high Nash welfare exists. It remains unknown whether similar results hold for the allocation of chores.

Besides EFX, there are other well-studied fairness notions, e.g., MMS [Budish, 2011] and PROPX [Moulin, 2019], for the allocation of chores. While it has been shown that MMS allocations are not guaranteed to exist for indivisible chores [Aziz *et al.*, 2017], there are many works that study its approximations [Aziz *et al.*, 2017; Aziz *et al.*, 2020a; Aziz *et al.*, 2019; Huang and Lu, 2021]. The state-of-the-art approximation ratio for MMS allocation for indivisible chores is $11/9$ by Huang and Lu [Huang and Lu, 2021]. Regarding PROPX allocations, in contrast to the allocation of goods, where PROPX allocation is not guaranteed to exist [Moulin, 2019; Aziz *et al.*, 2020b], it is shown that PROPX allocation always exists and can be computed efficiently for chores [Li *et al.*, 2022].

2 Preliminaries

We consider how to fairly allocate a set of m indivisible chores M to a group of n agents N. A bundle is a subset of items $X \subseteq M$. An allocation is represented by a n-partition $\mathbf{X} = (X_1, \cdots, X_n)$ of the items, where $X_i \cap X_j = \emptyset$ for all $i \neq j$ and $\cup_{i \in N} X_i = M$. In the allocation $\mathbf{X}$, agent $i \in N$ received bundle X_i. Each agent $i \in N$ has an additive cost function $c_i : 2^M \to \mathbb{R}^+ \cup \{0\}$. That is, for any $i \in N$ and $X \subseteq M$, $c_i(X) = \sum_{e \in X} c_i(\{e\})$.

When there is no confusion, we use c_{ie} and $c_i(e)$ to denote $c_i(\{e\})$ for convenience. Further, given any set $X \subseteq M$ and $e \in M$, we use $X + e$ and $X - e$ to denote $X \cup \{e\}$ and $X \setminus \{e\}$, respectively.

Definition 1 (α-EFX)**.** *For any $\alpha \geq 1$, an allocation $\mathbf{X}$ is α-approximate envy-free up to any item (α-EFX) if for any $i, j \in N$ and any $e \in X_i$,$c_i(X_i - e) \leq \alpha \cdot c_i(X_j)$. When $\alpha = 1$, the allocation $\mathbf{X}$ is EFX.*

Without loss of generality, we assume that all cost functions are normalized. That is, for any $i \in N$, $c_i(M) = 1$. Let $\sigma_i(j)$ be the j-th most costly item under c_i (with ties broken deterministically, e.g., by item ID). In other words, for every agent $i \in N$, we have $c_i(\sigma_i(1)) \geq c_i(\sigma_i(2)) \geq \cdots \geq c_i(\sigma_i(m))$. For each agent $i \in N$, we define $M_i^- = \{\sigma_i(j) : j \geq n\}$ as the set of *tail items*. Observe that we have $|M_i^-| = m - n + 1$ and $c_i(e) \geq c_i(e')$ for all $e \notin M_i^-$ and $e' \in M_i^-$. With this observation, we show that for every instance there exists a simple allocation that is $(m-n)$-EFX.

Lemma 2. *There exists a $(m-n)$-EFX allocation for every instance with m items and n agents.*

Proof. Fix an arbitrary agent, e.g., agent n, and define the allocation as follows. For all $i < n$, let $X_i = \{\sigma_n(i)\}$. Let $X_n = M_n^-$ be the set containing the remaining items. Obviously, the resulting allocation is EFX for all agents $i < n$ since $|X_i| = 1$. By the definition of M_n^-, for every $e \in M_n^-$, every $i < n$, we have

$$c_n(M_n^- - e) \leq (m-n) \cdot c_n(\sigma_n(i)) = (m-n) \cdot c_n(X_i)$$

Consequently, the allocation is $(m-n)$-EFX. □

Unfortunately the above approximation ratio is large, especially when m is large and n is small. In the following sections, we present algorithms that compute allocations with approximations of EFX that depend only on n. In particular, our algorithm computes a 5-EFX allocation when $n = 3$ and $3n^2$-EFX allocation for $n \geq 4$.

3 5-EFX Allocation for Three Agents

In this section we present the algorithm that computes a 5-EFX allocation when $n = 3$. The ideas we use to compute the allocation in this part inspires our design of approximation algorithm for general n.

Theorem 3. *There exists an algorithm that computes a 5-EFX allocation for three agents in $O(m \log m)$ time.*

Given an allocation $\mathbf{X}$, we say that agent i envies agent j if $c_i(X_i) > c_i(X_j)$; *strongly-envies* agent j if $\exists e \in X_i$ such that $c_i(X_i - e) > 5 \cdot c_i(X_j)$. In other words, an allocation is 5-EFX if and only if no agent strongly-envies another agent. For three agents we have $M_i^- = M \setminus \{\sigma_i(1), \sigma_i(2)\}$ for any $i \in N$. Observe that if there exists an agent $i \in N$ with $c_i(M_i^-) \leq 5 \cdot c_i(\sigma_i(2))$, then by allocating $\sigma_i(1)$ and $\sigma_i(2)$ to the other two agents and M_i^- to agent i, we end up having a 5-EFX allocation (following an analysis similar to the proof of Lemma 2). Hence it suffices to consider the case in which every agent $i \in N$ has $c_i(M_i^-) > 5 \cdot c_i(\sigma_i(2))$. In other words, item $\sigma_i(2)$ (and thus every item other than $\sigma_i(1)$) is "small" to agent i in the sense that it contributes at most a $1/6$ fraction to the total cost of $M - \sigma_i(1)$. Depending on whether $\sigma_i(1)$ has large cost, we classify the agents into two types: large agent and small agent.

Definition 4 (Large/Small Agent)**.** *We call agent $i \in N$ a* large *agent if $c_i(\sigma_i(1)) \geq 1/8$;* small *otherwise.*

The main intuition behind the definition is as follows. Recall that we are aiming for a 5-EFX allocation. For a large agent i, if $\sigma_i(1) \in X_j$ for some agent $j \neq i$, then as long as $c_i(X_i) \leq 5/8$, agent i does not strongly-envy agent j, even if $|X_j| = 1$. On the other hand, for a small agent i, every item $e \in M$ has cost $c_i(e) < 1/8$. Thus we can partition the items into bundles of roughly the same cost, under c_i, so that no matter which of these bundles agent i eventually receives, she will not strongly-envy any other agent.

Following the above intuitions, we proceed by considering how many small agents there are.

3.1 At Least Two Small Agents

W.l.o.g., suppose agent 1 and 2 are small. Agent 3 can be either small or large.

Lemma 5. *We can compute in polynomial-time a 3-partition (S_1, S_2, S_3) of M such that for both $i \in \{1, 2\}$, we have $c_i(S_j) \in [1/8, 5/8]$ for all $j \in \{1, 2, 3\}$.*

Note that Lemma 5 immediately implies Theorem 3 when there are at least two small agents for the following reasons. Since the costs of the three bundles S_1, S_2 and S_3 differ by a factor of at most 5, agent 1 and 2 will not strongly-envy any other agent as long as every agent gets exactly one bundle.

Algorithm 1: Sequential Placement

Initialize: $S_j \leftarrow \emptyset$ for all $j \in \{1,2,3\}$, $P \leftarrow M$ and $k \leftarrow 1$;
while $P \neq \emptyset$ **do**
  let $e^* \leftarrow \text{argmax}\{c_k(e) : e \in P\}$ and $j^* \leftarrow \text{argmin}\{c_k(S_j) : j \in \{1,2,3\}\}$;
  $S_{j^*} \leftarrow S_{j^*} + e^*, P \leftarrow P - e^*$;
  $k \leftarrow (k \bmod 2) + 1$;
Output: (S_1, S_2, S_3)

Therefore, by letting agent 3 pick her favourite bundle, i.e., the one with minimum $c_3(S_j)$, and allocating the remaining two bundles to agents 1 and 2 arbitrarily, we end up with a 5-EFX allocation: agent 3 does not envy agents 1 and 2; agents 1 and 2 do not strongly-envy any other agent.

Thus it remains to give the polynomial-time algorithm for the computation of (S_1, S_2, S_3). The main idea behind the algorithm is quite simple: since agents 1 and 2 have small costs on every item, round-robin like algorithms should work in computing such a partition.

The Algorithm. We initialize S_j as an empty bundle for all $j \in \{1,2,3\}$. Then we let agents 1 and 2 take turns to put the unallocated item with maximum cost into the bundle with smallest cost, both under her own cost function, until all items are allocated (refer to Algorithm 1).

It remains to show that for both $i \in \{1,2\}$, we have $c_i(S_j) \in [1/8, 5/8]$ for all $j \in \{1,2,3\}$. The complete analysis (and most of the lemmas to follow) can be found in the full version of the paper (see the supplementary material).

3.2 One Small Agent

Next we consider the case when there is exactly one small agent, say agent 3. Let $e_1 = \sigma_1(1)$ and $e_2 = \sigma_2(1)$ be the most costly item under c_1 and c_2, respectively. Since agents 1 and 2 are large, we have $c_1(e_1) \geq 1/8$ and $c_2(e_2) \geq 1/8$.

Lemma 6. *When there is exactly one small agent, there exists a 4-EFX allocation and could be found in polynomial time.*

Obviously there are two possibilities here, $e_1 = e_2$ and $e_1 \neq e_2$ respectively. For the first case, we can use the fact that the large item for agent 1 and 2 is the same, and assign it to agent 3. Let $X_3 = \{e_1\}$, and we compute an EFX allocation (X_1, X_2) between agents 1 and 2 on items $M - e_1$. In this allocation agent 3 obviously does not strongly-envy agents 1 and 2 because $|X_3| = 1$, and agents 1 and 2 do not envy each other by more than one item. On the other hand, since (X_1, X_2) is an EFX allocation on items $M - e_1$, by removing any item e from X_1 (resp. X_2), we have $c_1(X_1 - e) \leq 1/2$ (resp. $c_2(X_2 - e) \leq 1/2$). Since $c_1(X_3) = c_1(e_1) \geq 1/8$ and $c_2(X_3) = c_2(e_2) \geq 1/8$, the allocation is 4-EFX.

So it remains us to deal with the case that $e_1 \neq e_2$. And we use the fact that agent 3 is a small agent to compute an allocation that is fair to all.

The Algorithm. If $e_1 \neq e_2$, let $M^- = M - e_1 - e_2$. We first compute an EFX allocation (S_1, S_2, S_3) for three agents under cost function c_3, on items M^-. Assume w.l.o.g. that

Algorithm 2: Algorithm for 2 Large Agents

Initialize: $X_j \leftarrow \emptyset$ for all $j \in \{1,2,3\}$, $P \leftarrow M^-$;
Compute an EFX allocation (S_1, S_2, S_3) on items P under cost function c_3 ;
$X_1 \leftarrow S_1 + e_2$, $X_2 \leftarrow \text{argmin}\{c_2(S_2 + e_1), c_2(S_3)\}$;
$X_3 \leftarrow M - X_1 - X_2$;
Output: (X_1, X_2, X_3)

Algorithm 3: Algorithm for 3 Large Agents

Initialize: $X_j \leftarrow \emptyset$ for all $j \in \{1,2,3\}$, $P \leftarrow M^-$;
let $X_3 \leftarrow \{e_1, e_2\}$;
Compute an EFX allocation (S_1, S_2) on items P under cost function c_1 ;
let $S_{i^*} \leftarrow \text{argmin}\{c_3(S_1), c_3(S_2)\}$;
$X_2 \leftarrow \text{argmin}\{S_{i^*} + e_3, P - S_{i^*}\}$;
$X_1 \leftarrow P + e_3 - X_2$;
Output: (X_1, X_2, X_3)

$c_1(S_1) \leq c_1(S_2) \leq c_1(S_3)$, we decide the allocation $\mathbf{X}$ as follows. Let $X_1 = S_1 + e_2$, let agent 2 pick a bundle between $S_2 + e_1$ and S_3; then agent 3 gets the remaining bundle.

3.3 All Agents are Large

Finally, we consider the case when all agents are large. Let $e_1 = \sigma_1(1)$, $e_2 = \sigma_2(1)$ and $e_3 = \sigma_3(1)$. By definition we have $c_1(e_1) \geq 1/8$, $c_2(e_2) \geq 1/8$ and $c_3(e_3) \geq 1/8$.

Lemma 7. *When all three agents are large, there exists a 4-EFX allocation and could be found in polynomial time.*

As before, if there exist two of $\{e_1, e_2, e_3\}$ that are the same item, say $e_1 = e_2$, then we can easily compute an 4-EFX allocation by assigning e_1 to X_3 and computing an EFX allocation between agent 1 and 2 on the remaining items. Hence we assume e_1, e_2 and e_3 are three different items, and let $M^- = M \setminus \{e_1, e_2, e_3\}$.

The Algorithm. We initialize X_j as an empty bundle for all $j \in \{1,2,3\}$. We first assign both e_1 and e_2 to agent 3, and agent 3 quit the allocation. Then we compute an EFX allocation (S_1, S_2) on items M^- under cost function c_1. Assume w.l.o.g. that $c_3(S_1) \leq c_3(S_2)$. We let agent 2 pick a bundle between $S_1 + e_3$ and S_2, and assign the other bundle to agent 1 (refer to Algorithm 3).

We argue that above three algorithms run in $O(m \log m)$ time because under a fix cost function, computing an EFX allocation can be done by sorting items and allocating items sequentially to form a partition. Given this partition the final allocation can be determined in $O(1)$ time. In summary, in all cases we can compute a 5-EFX allocation for three agents in polynomial-time, which proves Theorem 3. From the above analysis we observe that it is crucial to distinguish whether an item e is large to an agent i: if it is, then by allocating the item to another agent j who values it small, we can eliminate the strong envyness from i to j; if it is not, then putting item e in X_i does not hurt the approximation ratio too much. In the following section, we show how these ideas can be extended to the general case when $n \geq 4$.

4 General Number of Agents

In this section, we give an algorithm that computes a $3n^2$-EFX allocation for any given instance with $n \geq 4$ agents.

Theorem 8. *There exists an algorithm that computes a $3n^2$-EFX allocation for any instance with n agents in $O(nm \log m)$ time.*

For each agent $i \in N$, we define $M_i^- = \{\sigma_i(j) : j \geq n\}$. As before (refer to Lemma 2 for a formal analysis), if there exists an agent i with $c_i(M_i^-) \leq 3n^2 \cdot c_i(\sigma_i(n-1))$, then we can easily compute a $3n^2$-EFX allocation by allocating exactly one item in $M \setminus M_i^-$ to each agent in $N - i$, and assigning the remaining items M_i^- to agent i. Since each agent other than i receives only one item, and agent i receives a set of items with total cost at most $3n^2$ times the cost of any other agent, the allocation is $3n^2$-EFX.

In the following we say that agent i strongly-envies agent j if for some $e \in X_i$ it holds $c_i(X_i - e) > 3n^2 \cdot c_i(X_j)$.

We define $b_i = \frac{1}{3n^2-n+2} \cdot c_i(M_i^-)$, and let $L_i = \{e \in M : c_i(e) \geq b_i\}$ be the set of *large* items of agent i. Note that by the above discussion we have $|L_i| \leq n-2$ for all $i \in N$. The main intuition behind the definition of large items is as follows. Our algorithm will compute an allocation $\mathbf{X} = (X_1, \ldots, X_n)$ ensuring that for each agent $i \in N$, either $X_i \cap L_i = \emptyset$, i.e., no large item in L_i is assigned to agent i; or $|X_i| = 1$. We show that as long as no large item is assigned to agent i, i does not strongly-envy any other agent that receives at least one item in L_i.

Lemma 9. *For any agent $i \in N$, if $X_i \cap L_i = \emptyset$ and $L_i \cap X_j \neq \emptyset$, then i does not strongly-envy agent j.*

Remark. A few difficulties arise when we try to extend the ideas we develop for three agents to general number of agents. First, for a large number of agents, it is no longer feasible to classify agents depending on how many large items they have because there are too many cases. Instead, our new algorithm removes the large items of each agent, and treats all agents as "small agents". Second, even if all agents have no large items, i.e., $L_i = \emptyset$ for all $i \in N$, it is not clear how to extend the Round Robin Placement algorithm (Lemma 5) to compute $O(n)$ subsets with upper and lower bounded costs for general number of agents. To get around this, we borrow existing results for the PROP1 allocation of goods, and show that when items are small to the agents, we can partition the items into $O(n)$ bundles such that the ratio between the costs of any two bundles is bounded by $3n^2$.

Let $L = \cup_{i \in N} L_i$ be the set of items that are large to at least one agent. Let $K = \cap_{i \in N} L_i$. Note that each item $e \in K$ is large to all agents. Let $M^- = M \setminus L$ be the set of items that are small to all agents.

Claim 4.1. *For each $i \in N$ and $e \in M^-$, we have $c_i(e) \leq \frac{1}{2n^2+2n} \cdot c_i(M^-)$.*

Recall that now we have three sets of items K, M^- and $L \setminus K$, each of which will be handled as follows.

- Since each item in K is large to all agents, we assign each of them to a unique agent chosen arbitrarily. Let N^* be these agents, and $N^- = N \setminus N^*$. Note that $|N^*| = |K|$. Our algorithm will not assign any further items to agents in N^*. Obviously these agents do not envy any other agents by more than one item. Moreover, if we can ensure that in the final allocation $X_i \cap L_i = \emptyset$ for all $i \in N^-$, then no agent strongly-envies any agent in N^*, by Lemma 9.
- By Claim 4.1, for all agent i, all items in M^- have small cost compared to $c_i(M^-)$. We show in Lemma 10 that we can partition the items in M^- into $|N^-|$ bundles evenly under every cost function of agents, and assign them to the agents in N^-. The key to the computation of the partition is to ensure that each bundle has a considerably large cost to every agent in N^-.
- It remains to assign items in $L \setminus K$. Recall that these are items that are large to some agent but not to all agents. Our algorithm assigns each $e \in L \setminus K$ to an arbitrary agent i for which $e \notin L_i$.

Lemma 10. *Given a set of items M^- and a group of agents N^- such that for all $i \in N^-$ and $e \in M^-$,$c_i(e) \leq \frac{1}{2n^2+2n} \cdot c_i(M^-)$, there exists a partition $(S_1, S_2, \ldots, S_{|N^-|})$ of M^- such that for all $i, j \in N^-$, $c_i(S_j) \geq \frac{1}{2n^2+2n} \cdot c_i(M^-)$.*

We show that no agent strongly-envies another agent. From the previous analysis, we know that agents in N^* do not envy any other agent by more than one item, and agents in N^- do not strongly-envy agents in N^*. It remains to show that agents in N^- do not strongly-envy each other. Recall that each agent $i \in N^-$ receives a bundle S_i (see Lemma 10), and possibly some other items from $L \setminus K$ that are small to agent i. By Lemma 10, for every $j \in N^-$ we have $c_i(X_j) \geq \frac{1}{2n^2+2n} \cdot c_i(M^-)$. Next we give an upper bound on $c_i(X_i)$. Recall that for any $i \in N$ we have $|L_i| \leq n-2$, and there are $(n-1)$ agents except i itself. Hence there are at most $(n-1) \cdot (n-2)$ items in $L \setminus K$ that are small to agent i, and

$$\begin{aligned} c_i(X_i) &\leq (1 - \frac{1}{2n^2+2n}) \cdot c_i(M^-) + (n-1) \cdot (n-2) \cdot b_i \\ &\leq \frac{2n^2+2n-1}{2n^2+2n} \cdot c_i(M^-) + \frac{n^2-3n+2}{2n^2+2n} \cdot c_i(M^-) \\ &\leq \frac{3n^2-n+1}{2n^2+2n} \cdot c_i(M^-) \leq \frac{3n^2}{2n^2+2n} \cdot c_i(M^-). \end{aligned}$$

Combining the upper bound on $c_i(X_i)$ and lower bound on $c_i(X_j)$, we have $c_i(X_i) \leq 3n^2 \cdot c_i(X_j)$ for any $i, j \in N^-$. Hence agents in N^- do not strongly-envy each other.

We finally argue that our algorithm for computing $3n^2$-EFX allocations for $n \geq 4$ agents takes $O(nm \log m)$ time. The main complexity comes from the division of items into three sets K, M^- and $L \setminus K$, which takes $O(nm \log m)$ time as each L_i can be computed in $O(m \log m)$ time. Given the three sets, it can be verified that allocating items in K, M^- and $L \setminus K$ takes $O(nm)$ time.

5 Bi-valued Instances

In this section, we consider the fair allocation problem with agents having bi-valued cost functions. That is, for any $i \in N$ and $e \in M$, there exist constants $a, b \geq 0$, $c_i(e) \in \{a, b\}$. Equivalently, for any $a \neq b$, we can scale the cost function so

that $c_i(e) \in \{\epsilon, 1\}$ where $\epsilon \in [0, 1)$. Note that when $\epsilon = 0$, the instance is a binary instance. We call the allocation $\mathbf{X}$ a *partial allocation* if $\cup_{i \in N} X_i \subsetneq M$. In the following, we present an algorithm that computes an EFX allocation when $n = 3$ and an algorithm that computes a $(n-1)$-EFX allocation when $n \geq 4$. We first give some definitions.

Definition 11 (Consistent Items)**.** *We call item $e \in M$ a* consistently large *item if for all $i \in N, c_i(e) = 1$; a* consistently small *item if for all $i \in N, c_i(e) = \epsilon$. All other items are called* inconsistent *items.*

Definition 12 (Large/small Items)**.** *We call item $e \in M$* large to agent i *if $c_i(e) = 1$;* small to i *if $c_i(e) = \epsilon$. Let $M_i^- = \{e \in M : c_i(e) = \epsilon\}$ be the items that are small to agent i. If an item is large (resp. small) to agent i, but is small (resp. large) to all other agents, we say that it is* large (resp. small) only to agent i.

The algorithm we use to compute an EFX allocation for three agents is based on the round-robin algorithm. The algorithm takes as input a set of items and an ordering of the agents $\{\sigma_1, \sigma_2, \ldots, \sigma_n\} = N$, and lets the agents pick their favourite item one-by-one, following the order σ, until all items are allocated. We call the output allocation $\mathbf{X}$ a round-robin allocation. Note that when $M' \neq M$, the allocation is partial. We index the rounds by $1, 2, \cdots, m$. Note that in each round, exactly one item is assigned. For every agent i, we denote that the last round she received an item by r_i.

Lemma 13. *For a given round-robin allocation, for any two agents $i \neq j$ such that $r_i < r_j$, agent i does not envy agent j, and agent j is EF1 towards agent i.*

Lemma 14. *Given the round-robin allocation on items M' and agents i and j with $r_i < r_j$, if there exists $e \in M \setminus M'$ such that $c_i(e) = \epsilon$ and $c_j(e) = 1$, then in the allocation $(\ldots, X_i + e, \ldots, X_j, \ldots)$, agent i is EFX towards agent j and agent j does not envy agent i.*

Theorem 15. *There exists an algorithm that computes an EFX allocation for three agents with bi-valued cost functions in $O(m \log m)$ time.*

Given an allocation $\mathbf{X}$, we say that agent i *strongly-envies* agent j if there exists $e \in X_i$ such that $c_i(X_i - e) > c_i(X_j)$. We first note that computing an EFX allocation is easy for some special cases. For instance, if there exist two agents sharing the same cost function, e.g., $c_1 = c_2$, we can compute an EFX allocation by partitioning the items into an EFX allocation under the cost function of agent 1, then let agent 3 pick her favourite bundle. Clearly, agent 3 would not envy the other two agents. In addition, since agents 1 and 2 share the same cost function, no matter which bundle they receive, they will not strongly-envy the other agents. Hence in the following we assume that all agents have different cost functions.

Overview. The main idea of our algorithm is to make use of Lemma 13 and 14 to construct an EFX allocation. In particular, given a round-robin allocation X, we would like to assign some unallocated items to improve the fairness guarantee to EFX. Note that for every agent i, every item can be categorised into one of the following four types: *consistently large, consistently small, large only to agent i, small only to agent i.* For example, suppose every agent i has an item e_i that is small only to agent i. Then by computing a round-robin allocation on items $M' = M \setminus \{e_1, e_2, e_3\}$ and allocating each e_i to agent i, the resulting allocation is EFX, by Lemma 14. When this does not hold, our algorithm carefully examines the number of items of each type in the viewpoint of each agent, and proceeds differently.

Lemma 16. *If there exists an item small only to some agent, then an EFX allocation can be computed in polynomial time.*

By Lemma 16, it remains to consider that case when every inconsistent item is large only to some agent i. In other words, suppose L_i is the set of items that are large only to agent i, then (L_1, L_2, L_3) is a partition of the inconsistent items into three sets. We finish the proof of Theorem 15 by the following two lemmas.

Lemma 17. *If there exists $|L_i| \geq 2$, then an EFX allocation can be computed in polynomial time.*

Lemma 18. *If $|L_i| \leq 1$ for all $i \in N$, then an EFX allocation can be computed in polynomial time.*

Finally, we consider instances with $n \geq 4$ agents and prove the following, whose proof can be found in the full version.

Theorem 19. *There exists an algorithm that computes a $(n-1)$-EFX allocation for $n \geq 4$ agents with bi-valued cost functions in $O(nm)$ time.*

6 Conclusion and Future Work

In this paper we propose algorithms that always compute a 5-EFX allocation for three agents and $3n^2$-EFX allocation for $n \geq 4$ agents. These are the first approximation ratios of EFX that are independent of m for the allocation of indivisible chores. Furthermore, we show that the approximation could be improved for bi-valued instances. We propose algorithms that always compute an EFX allocation for three agents with bi-valued cost functions and $(n-1)$-EFX allocation for $n \geq 4$ agents. There are many open problems regarding the computation of approximately EFX allocation for chores. For example, it would be interesting to investigate whether constant approximations of EFX allocation exist for general number of agents, and whether EFX allocations exist for three agents or any number of agents with bi-valued cost functions. Observe that to ensure (approximation of) EFX for an agent i, we often need to focus on increasing the costs of other agents, instead of minimizing the cost of i, which can possibly lead to inefficiency in the final allocation. It is thus interesting to study the existence of allocations for chores that are fair, e.g., approximation of EFX, MMS or PROPX, and efficient, e.g., Pareto optimal.

Acknowledgments

Xiaowei Wu is funded by FDCT (File no. 0143/2020/A3, SKL-IOTSC-2021-2023), the SRG of University of Macau (File no. SRG2020-00020-IOTSC) and GDST (2020B1212030003).

References

[Akrami *et al.*, 2021] Hannaneh Akrami, Bhaskar Ray Chaudhury, Kurt Mehlhorn, Golnoosh Shahkarami, and Quentin Vermande. Nash social welfare for 2-value instances. *CoRR*, abs/2106.14816, 2021.

[Alon, 1987] Noga Alon. Splitting necklaces. *Advances in Mathematics*, 63(3):247–253, 1987.

[Amanatidis *et al.*, 2020] Georgios Amanatidis, Evangelos Markakis, and Apostolos Ntokos. Multiple birds with one stone: Beating 1/2 for EFX and GMMS via envy cycle elimination. *Theor. Comput. Sci.*, 841:94–109, 2020.

[Amanatidis *et al.*, 2021] Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, Alexandros Hollender, and Alexandros A. Voudouris. Maximum nash welfare and other stories about EFX. *Theor. Comput. Sci.*, 863:69–85, 2021.

[Aziz and Brown, 2020] Haris Aziz and Ethan Brown. Random assignment under bi-valued utilities: Analyzing hylland-zeckhauser, nash-bargaining, and other rules. *arXiv e-prints*, pages arXiv–2006, 2020.

[Aziz and Mackenzie, 2016] Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for any number of agents. In *FOCS*, pages 416–427. IEEE Computer Society, 2016.

[Aziz *et al.*, 2017] Haris Aziz, Gerhard Rauchecker, Guido Schryen, and Toby Walsh. Algorithms for max-min share fair allocation of indivisible chores. In *AAAI*, pages 335–341. AAAI Press, 2017.

[Aziz *et al.*, 2019] Haris Aziz, Bo Li, and Xiaowei Wu. Strategyproof and approximately maxmin fair share allocation of chores. In *IJCAI*, pages 60–66. ijcai.org, 2019.

[Aziz *et al.*, 2020a] Haris Aziz, Bo Li, and Xiaowei Wu. Approximate and strategyproof maximin share allocation of chores with ordinal preferences. *CoRR*, abs/2012.13884, 2020.

[Aziz *et al.*, 2020b] Haris Aziz, Hervé Moulin, and Fedor Sandomirskiy. A polynomial-time algorithm for computing a pareto optimal and almost proportional allocation. *Oper. Res. Lett.*, 48(5):573–578, 2020.

[Berger *et al.*, 2021] Ben Berger, Avi Cohen, Michal Feldman, and Amos Fiat. (almost full) EFX exists for four agents (and beyond). *CoRR*, abs/2102.10654, 2021.

[Bhaskar *et al.*, 2021] Umang Bhaskar, A. R. Sricharan, and Rohit Vaish. On approximate envy-freeness for indivisible chores and mixed resources. In *APPROX-RANDOM*, volume 207 of *LIPIcs*, pages 1:1–1:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[Budish, 2011] Eric Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011.

[Caragiannis *et al.*, 2019] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. *ACM Trans. Economics and Comput.*, 7(3):12:1–12:32, 2019.

[Chan *et al.*, 2019] Hau Chan, Jing Chen, Bo Li, and Xiaowei Wu. Maximin-aware allocations of indivisible goods. In *AAMAS*, pages 1871–1873. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[Chaudhury *et al.*, 2020a] Bhaskar Ray Chaudhury, Jugal Garg, and Kurt Mehlhorn. EFX exists for three agents. In *EC*, pages 1–19. ACM, 2020.

[Chaudhury *et al.*, 2020b] Bhaskar Ray Chaudhury, Telikepalli Kavitha, Kurt Mehlhorn, and Alkmini Sgouritsa. A little charity guarantees almost envy-freeness. In *SODA*, pages 2658–2672. SIAM, 2020.

[Chaudhury *et al.*, 2021] Bhaskar Ray Chaudhury, Jugal Garg, Kurt Mehlhorn, Ruta Mehta, and Pranabendu Misra. Improving EFX guarantees through rainbow cycle number. In *EC*, pages 310–311. ACM, 2021.

[Ebadian *et al.*, 2021] Soroush Ebadian, Dominik Peters, and Nisarg Shah. How to fairly allocate easy and difficult chores. *CoRR*, abs/2110.11285, 2021.

[Edward Su, 1999] Francis Edward Su. Rental harmony: Sperner's lemma in fair division. *The American mathematical monthly*, 106(10):930–942, 1999.

[Foley, 1967] Duncan Foley. Resource allocation and the public sector. *Yale Economic Essays*, pages 45–98, 1967.

[Gafni *et al.*, 2021] Yotam Gafni, Xin Huang, Ron Lavi, and Inbal Talgam-Cohen. Unified fair allocation of goods and chores via copies. *CoRR*, abs/2109.08671, 2021.

[Garg and Murhekar, 2021] Jugal Garg and Aniket Murhekar. Computing fair and efficient allocations with few utility values. In *SAGT*, volume 12885 of *Lecture Notes in Computer Science*, pages 345–359. Springer, 2021.

[Garg *et al.*, 2021] Jugal Garg, Aniket Murhekar, and John Qin. Fair and efficient allocations of chores under bivalued preferences. *CoRR*, abs/2110.09601, 2021.

[Huang and Lu, 2021] Xin Huang and Pinyan Lu. An algorithmic framework for approximating maximin share allocation of chores. In *EC*, pages 630–631. ACM, 2021.

[Li *et al.*, 2022] Bo Li, Yingkai Li, and Xiaowei Wu. Almost (weighted) proportional allocations for indivisible chores. In *WWW*, pages 122–131. ACM, 2022.

[Lipton *et al.*, 2004] Richard J. Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On approximately fair allocations of indivisible goods. In *EC*, pages 125–131. ACM, 2004.

[Moulin, 2019] Hervé Moulin. Fair division in the internet age. *Annual Review of Economics*, 11(1):407–441, 2019.

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

[Steinhaus, 1948] Hugo Steinhaus. The problem of fair division. *Econometrica*, 16:101–104, 1948.