Highlights

Almost Proportional Allocations of Indivisible Chores: Computation, Approximation and Efficiency[1]

Haris Aziz[⋆], Bo Li[#], Hervé Moulin[†], Xiaowei Wu[‡], Xinran Zhu[⋆]

[⋆]UNSW Sydney, haris.aziz@unsw.edu.au, zxrzita@gmail.com
[#]The Hong Kong Polytechnic University, comp-bo.li@polyu.edu.hk
[†]University of Glasgow, herve.moulin@glasgow.ac.uk
[‡]University of Macau, xiaoweiwu@um.edu.mo

- We design algorithms to compute PROPX allocations for indivisible chores when the agents have asymmetric weights.
- We design algorithms, which only access the agents' ordinal preferences, to compute approximate PROPX allocations.
- We prove the incompatibility of PROPX and Pareto optimality and identify some special cases when a Pareto optimal and PROPX allocation exists.

[1]Some results have appeared in The ACM Web Conference 2022 (WWW'22) [52].

Almost Proportional Allocations of Indivisible Chores: Computation, Approximation and Efficiency[1]

Haris Aziz[⋆], Bo Li[#], Hervé Moulin[†], Xiaowei Wu[‡], Xinran Zhu[⋆]

[⋆]UNSW Sydney, haris.aziz@unsw.edu.au, zxrzita@gmail.com
[#]The Hong Kong Polytechnic University, comp-bo.li@polyu.edu.hk
[†]University of Glasgow, herve.moulin@glasgow.ac.uk
[‡]University of Macau, xiaoweiwu@um.edu.mo

Abstract

Proportionality (PROP) is one of the simplest and most intuitive fairness criteria used for allocating items among agents with additive utilities. However, when the items are indivisible, ensuring PROP becomes unattainable, leading to increased focus on its relaxations. In this paper, we focus on the relaxation of proportionality up to any item (PROPX), where proportionality is satisfied if an arbitrary item is removed from every agent's allocation. We show that PROPX is an appealing fairness notion for the allocation of indivisible chores, which approximately implies some share-based notions, such as maximin share (MMS) and AnyPrice share (APS). We further provide a comprehensive understanding of PROPX allocations, regarding the computation, approximation, and compatibility with efficiency. On top of these, we extend the study to scenarios where agents do not share equal liability towards the chores, and approximate PROPX allocations using partial information about agents' utilities.

Keywords: Indivisible Chores, PROPX, Ordinal Preferences, Pareto Optimality

[1]Some results have appeared in The ACM Web Conference 2022 (WWW'22) [52].

1. Introduction

Proportionality (PROP), formally introduced 75 years ago by the mathematician Steinhaus [57] to divide a non atomic cake – or any other infinitely divisible item like land, clean water or money – is the simplest and most intuitive test of fairness: my share should be worth at least $\frac{1}{n}$-th of the total value of the cake I am sharing with $n-1$ other participants; this is the best worst-case share that can be secured simultaneously for everyone. Fast forwarding to the 21st century, a growing literature developed mostly by computer scientists together with mathematicians, economists, and other social scientists, focuses on the fair moneyless allocation of indivisible items, such as jobs, offices, seats in school, or immigration visas [54, 3]. Providing each agent with a reasonably high worst-case share remains a central fairness concern, but PROP is no longer feasible in its pristine form as we see when trying to divide one diamond and several worthless rocks: only an approximate version of PROP will work.

Most of the attention turned instead to the intuitive and elegant concept of the maximin share (MMS) guarantee [28]: the worth of my least valuable share in the best n-partition of the objects I can choose. But MMS is not always feasible either[2] and it too must be approximated; moreover, unlike PROP or its two approximations below, the MMS test (or its factor approximation [1, 2, 49]) is not practical because recognizing that a given allocation of the items passes the test requires the NP-hard task of computing the MMS utility.

The first approximation of the PROP test, written as PROP1 [34], is relatively weak as we explain below. For the allocation of goods (desirable items that can be freely disposed of), it says that every agent's share achieves the original proportionality after *adding* one carefully chosen item outside the agent's share. For the allocation of chores (also known as bads, that generate *disutility* and cannot be discarded), the same obtains by *subtracting* one well-chosen item from the share. In both cases PROP1 is compatible with an efficient (Pareto optimal) division of the items [21, 11]. The weakness is that PROP1 follows from the substantially stronger property EF1, i.e., the similar approximation of the powerful envy freeness test[3] by adding a good

[2] Although the instances proving this are considerably more complex than the diamond and rocks example: [51, 12, 38].

[3] I prefer my share to the share of any other agent.

or subtracting a chore [53]. So for goods, EF1 overshadows PROP1 because it is also compatible with efficiency [31, 21] and equally easy to verify. For chores, the compatibility between EF1 and efficiency remains unknown and turns out to be a challenging problem [35, 40, 41].

The next approximation, PROPX is the considerably stronger requirement that we achieve PROP by adding *any* good outside the agent's share or subtracting *any* chore from this share. Also, when we divide goods a PROPX allocation may not exist even in fairly simple problems.[4] Surprisingly the situation is much more favorable for chores, where PROPX allocations always exist and can be computed by fairly simple algorithms [54, 52]. The goal of this paper is to deepen our understanding of PROPX which we regard as one of the most appealing fairness requirements for the allocation of indivisible chores among agents endowed with additive utilities.

We also discuss two variants of this standard model. In the first one agents do not have the same liability toward chores; for example, people in leadership positions versus lower-level employees in a company, teaching obligations differ for university staff with different status and countries have different responsibilities to reduce CO2 emissions based on their populations or past actions. This is referred to as the *asymmetric* or *weighted* setting in our work. Weighted fairness has been well justified since the 1990s in the context of the cake-cutting problem [56], and is recently adapted to the relaxations of EF [32, 58] and MMS [36, 9]. Next, we note that in practical applications with a large number of agents and/or chores, we do not expect all agents to form full-fledged cardinal evaluations of these items, to say nothing of the potential computational complexity due to the large sizes, therefore it is important to adapt the analysis based on partial information such as the ordinal ranking of the chores. Then our problem is to investigate the extent we are able to approximate PROPX using partial information.

1.1. Main Results

We study the problem of allocating m indivisible chores to n agents, where each agent i has an obligation $s_i \geq 0$ on the share of chores she needs to finish, and $s_1 + \cdots + s_n = 1$. Based on whether all agents have share $1/n$, we call them symmetric (unweighted) or asymmetric (weighted). Informally, an allocation

[4] A simple example has 3 agents with identical utilities $(3, 3, 3, 3, 1)$ over five goods, where one of the three agents must have value no greater than 3 but her proportional share is $\frac{13}{3} > 3$.

is called PROPX if for any agent, by removing an arbitrary item from her bundle, her cost is no more than her share in the system. We argue that for indivisible chores, PROPX may be a more reliable relaxation of (weighted) proportionality than MMS and APS, where the definition of APS is deferred to Definition 4.2. This is because the existence of MMS/APS allocations is not guaranteed even with three symmetric agents. However, we show that (weighted) PROPX allocations always exist. Moreover, any (weighted) PROPX allocation ensures 2-approximation of MMS for symmetric agents and of APS for asymmetric agents; however, an arbitrary MMS or APS allocation can be as bad as $\Theta(n)$-approximation regarding PROPX.

As a warm-up, we first show that the *top-trading envy-cycle elimination* [26] algorithm computes a PROPX allocation when the agents are symmetric when the agents have the same ordinal preference for the items (which is named IDO instances in [48], short for identical ordering). Using the techniques from [27, 19, 48], the algorithm can be converted to handle general non-IDO instances easily. Then we move to the general case when the agents have arbitrary shares. Our algorithm *bid-and-take* takes the efficiency of the allocation into consideration – each item, from the highest to the lowest cost, is allocated to the agent who has the smallest cost on that item.

Result 1 (Lemma 4.2 and Theorem 4.1)**.** *The allocation returned by the bid-and-take algorithm is (weighted) PROPX and 2-approximate APS.*

Algorithm bid-and-take actually guarantees the tight approximation ratio to the optimal social cost subject to the (weighted) PROPX constraint, i.e., the *price of fairness* [25, 30]. In the appendix, we show that the tight bound for the price of fairness regarding PROPX is $\Theta(n)$ for the unweighted case, $\Theta(m)$ for the weighted IDO case and unbounded for the weighted case.

Following the recent works on the partial information setting [5, 43, 10], we study the problem of designing algorithms that only use agents' ordinal preferences without exact cardinal values. The intuition behind our algorithms (see Algorithms 3 and 4) is as follows. We partition agents into two sets so that each agent in the first set gets a single but large item, then a standard algorithm, such as the (weighted) round-robin, is called on the agents in the second half to evenly allocate the remaining small items. Although the idea of splitting agents into two parts looks artificial, the approximation ratio turns out to be optimal. A by-product result in this part is that weighted EF1 allocations exist for IDO instances.

Result 2 (Lemma 5.1 and Theorem 5.2)**.** *With ordinal preferences, our al-*

gorithms achieve 2-approximate (weighted) PROPX for both symmetric and asymmetric agents. Moreover, the approximation ratio is optimal: no algorithm can achieve a better-than-2 approximation using only ordinal information, even for symmetric agents.

Last but not least, we investigate the compatibility between PROPX and Pareto optimality (PO). We first prove that PROPX and PO are not compatible when there are items that are zero-valued by some agents. When all items are positively valued by all agents[5], PROPX and fPO are still not compatible, in contrast with [11] where fPO and PROP1 allocations exist for chores. We then identify some cases where we can obtain PROPX and PO allocations: (1) two agents with any additive valuations and symmetric weights, and (2) any number of agents with lexicographic or bi-valued valuations, even if agents may have asymmetric weights. The instances when agents have lexicographic valuations have been widely considered in the literature [22, 46, 7, 47]. The case when agents have bi-valued utility functions [4] is also one of the most well-studied restricted cases of the fair division problem. Our result aligns with [35, 40] which shows the compatibility of EF1 and PO under bi-valued valuations. As justified therein, the study of bi-valued valuations is of practical interest since reporting exact numerical utilities can be cumbersome for agents in many real-world scenarios. We leave the compatibility between PROPX and PO when the valuations are generally additive and all items have positive costs by all agents as an open problem.

Result 3 (Theorem 6.1, Propositions 6.2 and 6.3)**.** *PROPX and PO are not compatible in general. If all items are positively valued by all agents, PROPX and PO allocations exist when (1) there are two agents with any additive valuations and symmetric weights, or (2) any number of agents with lexicographic or bi-valued valuations, even if agents may have asymmetric weights.*

[5]This is actually the general setting if we consider a weaker definition of PROPX where the removed item must have a non-zero cost to the agent. A similar weaker definition of EFX is discussed in [55, 31]. Under the weaker definition, if there is an item for which an agent has zero cost, we can allocate this item to that agent, which will not violate the requirement anyway. Therefore, without loss of generality, we can assume all the items are positively valued by all agents.

1.2. Other Related Works

EF1 and EFX Allocations. Two widely studied relaxations of envy-freeness are *envy-freeness up to one item* (EF1) [53] and *envy-freeness up to any item* (EFX) [31], which were first proposed for goods. Informally, EF1 allocations require that the envy is eliminated after removing one item from a bundle. On the positive side, Lipton et al. [53] and Bhaskar et al. [26] proved that EF1 allocations are guaranteed to exist and can be found efficiently even when the agents have monotone combinatorial valuations. On the negative side, some EF1 allocations can be unfair even when the instance admits a fairer allocation. For example, consider allocating one large good and two identical small goods among two agents; if one agent gets a small good and the other agent gets the remaining goods, the allocation is EF1. EFX is proposed to improve the fairness guarantee, where envy is eliminated after removing any item from the bundle. Thus in the previous example, an EFX allocation will allocate both small goods to the agent who does not obtain the large one. Though EFX is fairer, it is still unknown whether such allocations are guaranteed to exist or not, except for some special cases [55, 33, 24, 6]. Particularly, Plaut and Roughgarden [55] proved that EFX allocations exist for IDO instances with goods. Our work complements this result by showing EFX allocations also exist for IDO instances with chores. Recently, Baklanov et al. [16] proved the existence of PROPm allocations, which is a criterion sitting somewhere between PROP1 and PROPX.

MMS and PROPm Allocations. Besides PROP1 and PROPX, MaxMinShare (MMS) guarantee [28] is a well-studied relaxation of proportionality. On one hand, it has been shown that for both goods [51] and chores [12], MMS allocations are not guaranteed to exist. Stronger inapproximability results are proved in [38]. On the other hand, a series of constant approximation algorithms are designed in the last decade, e.g., see [51, 36, 42, 48], with the best-known approximation ratio being $3/4 + 3/3836$ for goods [1] and $13/11$ for chores [49]. For goods, while PROPX allocations may not exist, Baklanov et al. [15] proposed *proportionality up to the maximin item* (PROPm), which is a criterion sitting somewhere between PROP1 and PROPX. In a subsequent work [16], the existence of PROPm allocations is proved.

Weighted Fairness. While most literature studies the special yet important case where agents have equal entitlement or obligation share to the items, there is also fast-growing recent literature on the more general model in

which agents may have arbitrary and possibly unequal shares. For example, Farhadi et al. [36] and Aziz et al. [9] adapted MMS to this setting for goods and chores, respectively and designed approximation algorithms accordingly. Babaioff et al. [13, 14] provided different generalizations of MMS to this case. Weighted EF1 allocations are known to exist and can be computed efficiently for goods [32] and for chores [58]. Recently, AnyPrice Share (APS) fairness was introduced by Babaioff et al. [13], where a 3/5-approximation algorithm is designed for goods and a 2-approximation algorithm is designed for chores. Recently, the approximation for chores is improved to 1.733 by Feige and Huang [37].

Partial Information. All the aforementioned fairness notions related to EF and PROP are defined using agents' cardinal preferences. However, in practical applications, the numbers of agents and items can be very large, and thus it may be impractical and even impossible for the algorithm to collect the complete information on all these cardinal values. Accordingly, one line of research in the literature studies how to use partial information on the preferences to compute approximately fair allocations. Ordinal information setting is a typical case where the algorithm only knows each agent's ranking over the items without cardinal values. For goods, using ordinal preferences to compute approximately MMS allocations has been studied in [5, 43] and the optimal approximation is logarithmic. For chores, Aziz et al. [10] showed that the tight approximation ratio is between 1.405 and 5/3, and recently, the upper bound is improved to 8/5 by Feige and Huang [37]. In our work, we aim to explore the limit of ordinal preferences to approximate PROPX fairness.

Fairness vs. Efficiency. Besides fairness, efficiency, which is a competing criterion to fairness, is another important criterion to evaluate allocations. One of the most intriguing problems is to understand if these fairness notions can be satisfied together with PO and fPO. While the compatibility between fairness and efficiency for goods is well understood [21, 39, 20], the problem for chores is not. For indivisible chores, it is shown in [35] and [40] that when agents have bi-valued valuations, or when there are at most two agents, EF1 and PO allocations exist and can be found in polynomial time. Recently, Garg et al. [41] extended these results to the cases when there are three agents and when there are at most two cost functions. Another efficiency-fairness tradeoff question is to understand the loss in social welfare when

fairness is enforced, which is quantitatively measured by *price of fairness*. Bounding the price of fairness for goods and chores is widely studied in the literature [25, 30, 44, 23, 17, 45]. In the appendix of this paper, we study the price of fairness for indivisible chores under (weighted) PROPX requirement and show that our algorithm achieves the optimal ratio.

1.3. Road Map

The following sections are organized as follows. In Section 2, we introduce the formal definitions of our problem. In Sections 3 and 4, we design algorithms to compute PROPX allocations for the symmetric and asymmetric settings. In Section 5, we consider the ordinal setting when we only know the rankings of the agents and design algorithms to compute approximately PROPX allocations. Finally, we consider the compatibility between PROPX and PO in Section 6. Some missing proofs and the discussion of the price of fairness regarding PROPX can be found in the appendix.

2. Model and Solution Concepts

We consider the problem of fairly allocating a set of m indivisible chores M to a group of n agents N. Each agent $i \in N$ has a cost function $c_i : 2^M \to \mathbb{R}^+ \cup \{0\}$. The cost functions are assumed to be additive in the current work; that is, for any item set $S \subseteq M$, $c_i(S) = \sum_{e \in S} c_i(\{e\})$. When there is no confusion, we use $c_i(e)$ to denote $c_i(\{e\})$. Since we consider proportional fairness in this work, we sometimes assume without loss of generality that the cost functions are normalized, i.e., $c_i(M) = 1$. An allocation is represented by a partition of the items $\mathbf{X} = (X_1, \ldots, X_n)$, where each agent i obtains X_i, $X_i \cap X_j = \emptyset$ for all $i \neq j$ and $\cup_{i \in N} X_i = M$. An allocation is called partial if $\cup_{i \in N} X_i \neq M$. Let the social cost of the allocation $\mathbf{X}$ be $\mathrm{sc}(\mathbf{X}) = \sum_{i \in N} c_i(X_i)$. If some item is fractionally allocated to more than one agent, the allocation is called *fractional* and is denoted by $x = (x_1, \ldots, x_n)$ where $x_i = (x_{i,1}, \cdots, x_{i,m})$ is the allocation of agent i and $0 \leq x_{i,e} \leq 1$ is the fraction of item e given to agent i. Note that it is required that $\sum_{i \in N} x_{i,e} = 1$ for each item $e \in M$. A fractional allocation y Pareto improves a fractional allocation x if $c_i(y_i) \leq c_i(x_i)$ for all $i \in N$ and for some i the inequality is strict. We will call an allocation *Pareto optimal* (PO), if no integral allocation Pareto improves it. An allocation that cannot be Pareto improved by any fractional allocation is called *fractional Pareto optimal* (fPO). Clearly, an fPO allocation is PO as well. We are sometimes interested in a special setting,

identical ordering (IDO), in which all agents agree on the same ranking of the items, i.e., $c_i(e_1) \geq \cdots \geq c_i(e_m)$ for all $i \in N$. Note that in an IDO instance, the agents' cardinal cost functions can still be different.

We next define *envy-freeness*, *proportionality* and their relaxations. For ease of discussion, in this section, we focus on the case of symmetric agents, also called *unweighted case*, and defer the definitions for asymmetric-agent case to Section 4.

Definition 2.1 (EF and PROP)**.** *An allocation* $\mathbf{X}$ *is envy-free (EF) if* $c_i(X_i) \leq c_i(X_j)$ *for any* $i, j \in N$*. The allocation is proportional (PROP) if* $c_i(X_i) \leq \mathsf{PROP}_i$ *for any* $i \in N$*, where* $\mathsf{PROP}_i = (1/n) \cdot c_i(M)$ *is agent* i*'s proportional share for all items.*

For normalized cost functions $\mathsf{PROP}_i = 1/n$, $\forall i \in N$.

Definition 2.2 (EF1 and EFX)**.** *An allocation* $\mathbf{X}$ *is envy-free up to one item (EF1) if for any* $i, j \in N$*, there exists* $e \in X_i$ *such that* $c_i(X_i \setminus \{e\}) \leq c_i(X_j)$*. The allocation is envy-free up to any item (EFX) if for any* $i, j \in N$ *and any* $e \in X_i$*,* $c_i(X_i \setminus \{e\}) \leq c_i(X_j)$*.*

It is easy to see that any EFX allocation is EF1, but not vice versa. We adopt similar ideas to relax the definition of proportionality.

Definition 2.3 (PROP1 and PROPX)**.** *For any* $\alpha \geq 1$*, an allocation* $\mathbf{X}$ *is* α*-approximate proportional up to one item (*α*-PROP1) if for any* $i \in N$*, there exists* $e \in X_i$ *such that* $c_i(X_i \setminus \{e\}) \leq \alpha \cdot \mathsf{PROP}_i$*. The allocation is* α*-approximate proportional up to any item (*α*-PROPX) if for any* $i \in N$ *and any* $e \in X_i$*,* $c_i(X_i \setminus \{e\}) \leq \alpha \cdot \mathsf{PROP}_i$*. When* $\alpha = 1$*, allocation* $\mathbf{X}$ *is PROP1 or PROPX, respectively.*

Similarly, any PROPX allocation is PROP1, but not vice versa. As we will see for any additive cost functions, PROPX allocations exist and can be found in polynomial time. Thus we always focus on PROPX allocations in this work. Next, we show that PROPX is weaker than EFX.

Lemma 2.1. *Any EFX allocation is PROPX.*

Proof. For any EFX allocation $\mathbf{X}$ and any agent i, $c_i(X_i \setminus \{e\}) \leq c_i(X_j)$ for all $e \in X_i$ and $j \in N$. Summing up the inequalities for all j, we have $n \cdot c_i(X_i \setminus \{e\}) \leq c_i(M)$. Thus $\mathbf{X}$ is PROPX. □

Finally, we recall the definition of maximin share fairness.

Definition 2.4 (MMS)**.** *Let $\Pi(M)$ be the set of all n-partitions of M. For any agent $i \in N$, her maximin share (MMS) is defined as*

$$\mathsf{MMS}_i = \min_{\mathbf{X} \in \Pi(M)} \max_{j \in N} \{c_i(X_j)\}.$$

For any $\alpha \geq 1$, an allocation $\mathbf{X}$ is α-approximate maximin share fair (α-MMS) if $c_i(X_i) \leq \alpha \cdot \mathsf{MMS}_i$ for all $i \in N$. When $\alpha = 1$, allocation $\mathbf{X}$ is MMS fair.

Given the definition of MMS fairness, it is not hard to observe the following inequality.

$$\mathsf{MMS}_i \geq \max \left\{ \max_{e \in M} \{c_i(e)\}, \mathsf{PROP}_i \right\}, \forall i \in N. \tag{1}$$

Lemma 2.2. *Any PROPX allocation is 2-MMS.*

Proof. We will prove a stronger argument here: for any PROPX allocation $\mathbf{X}$ and for any agent i, either $|X_i| \leq 1$ or $c_i(X_i) \leq 2 \cdot \mathsf{PROP}_i$. Then by Equation (1), Lemma 2.2 holds. For any agent i, if $|X_i| \leq 1$, the claim holds trivially. If $|X_i| \geq 2$, letting $e_i = \arg\min_{e \in X_i} \{c_i(e)\}$, we have

$$c_i(X_i \setminus \{e_i\}) \leq \mathsf{PROP}_i,$$

and

$$c_i(e_i) \leq c_i(X_i \setminus \{e_i\}) \leq \mathsf{PROP}_i.$$

Thus $c_i(X_i) = c_i(X_i \setminus \{e_i\}) + c_i(e_i) \leq 2 \cdot \mathsf{PROP}_i$. □

The approximation ratio in the lemma is tight. Consider an instance with two identical agents and two identical items. Allocating both items to one of them is PROPX but only 2-MMS.

Lemma 2.3. *There exists an MMS allocation that is $\Theta(n)$-PROPX.*

Proof. Consider an instance with n agents and $m = n$ items where n is sufficiently large. In this instance, all agents have identical cost functions for the items. For each agent i, let $c_i(e_1) = n - 1$ and $c_i(e_j) = 1$ for all $j = 2, \ldots, n$. Thus

$$\mathsf{MMS}_i = n - 1 \quad \text{and} \quad \mathsf{PROP}_i = \frac{2(n-1)}{n}.$$

Consider an allocation where $X_i = \{e_2, \ldots, e_n\}$ is allocated to some agent i. Note that $c_i(X_i) = n - 1 = \mathsf{MMS}_i$. However, this allocation is not fair regarding PROPX because $c_i(X_i \setminus \{e\}) = n - 2 = \Theta(n) \cdot \mathsf{PROP}_i$ for any $e \in X_i$. □

Actually, no allocation can be worse than n-PROPX, as for any chore allocation instance, the most unfair allocation is to allocate all items to a single agent, which is n-PROPX.

3. Warm-up: Unweighted Agents

The existence of PROPX allocations for chores was first proved by Moulin [54] via a novel algorithm with running time $O(nm^2)$. In this section, we show that PROPX allocations can also be obtained by the commonly used techniques, namely, envy-cycle elimination and IDO reduction. We use envy-cycle elimination algorithm to compute PROPX allocations for IDO instances in this section, and defer the IDO reduction to handle the general cost functions to Section 4.

The Algorithm. The envy cycle elimination algorithm was first proposed in [53] for goods and was adapted to chores in [26] recently. Given any (partial) allocation $\mathbf{X} = (X_1, \ldots, X_n)$, we say that agent i *envies* j if $c_i(X_i) > c_i(X_j)$ and *most-envies* j if $c_i(X_i) > c_i(X_j)$ and $j \in \arg\min_{k \in N} c_i(X_k)$. Note that an agent may have multiple most-envied agents. For any allocation $\mathbf{X}$, we can construct a directed graph G_X, called the *top-trading envy graph*, where the agents are nodes and there is a directed edge from i to j if and only if i most-envies j. A directed cycle $C = (i_1, \ldots, i_d)$ is referred as a *top-envy cycle*. For any top-envy cycle C, the *cycle-swapped allocation* $\mathbf{X}^C$ is obtained by reallocating bundles backwards along the cycle. That is, $X_i^C = X_i$ if i is not in C, and

$$X_{i_j}^C = \begin{cases} X_{i_{j+1}} & \text{for all } 1 \le j \le d-1 \\ X_{i_1} & \text{for } j = d. \end{cases} \tag{2}$$

The algorithm works by assigning, at each step, the unassigned item *with the highest cost* to an agent who does not envy anyone else (i.e., a non-envious agent who is a "sink" node in the top-trading envy graph). If the top-trading envy graph G_X does not have a sink, then it must have a cycle

[26]. Then resolving the top-trading envy cycles, by executing the corresponding cycle-swapped allocation, guarantees the existence of a sink agent in the top-trading envy graph. The full description of the algorithm is introduced in Algorithm 1. The following example demonstrates the execution of Algorithm 1, which also shows that the returned allocation may not be PO.

Example 3.1. *Consider an instance with two agents and three items. Note that the costs are shown in the following table.*

	$c_i(e_1)$	$c_i(e_2)$	$c_i(e_3)$
Agent 1	$\boxed{0.4}$	0.3	0.3
Agent 2	0.4	$\boxed{0.4}$	$\boxed{0.2}$

Algorithm 1 returns an allocation $\mathbf{X}$ *as shown in squares. Note that* $\mathbf{X}$ *is not PO since it is Pareto dominated by allocation* $\mathbf{X}'$ *with* $X'_1 = \{e_2\}$ *and* $X'_2 = \{e_1, e_3\}$.

Algorithm 1: Top-trading Envy Cycle Elimination

Input: IDO instance with $c_i(e_1) \geq \cdots \geq c_i(e_m)$ for all $i \in N$.
Initialize: $\mathbf{X} = (X_1, \ldots, X_n)$ where $X_i \leftarrow \emptyset$ for all $i \in N$.
for $j = 1, 2, \ldots, m$ **do**
    if *there is no sink in* G_X **then**
        Let C be any cycle in G_X.
        Reallocate the items according to X^C (i.e., the cycle-swapped allocation) and update the allocation $\mathbf{X}$.
    Choose a sink k in the graph G_X and update $X_k \leftarrow X_k \cup \{e_j\}$ according to Equation (2).
Output: Allocation $\mathbf{X} = (X_1, \ldots, X_n)$.

Algorithm 1 is the same as the one designed in [18, 26] except that we allocate the item with the highest cost to the sink agent at each step. It is proved in [26] that no matter which item is allocated, the returned allocation is EF1. In the following, we show a stronger argument: if we select the item with the highest cost at each step, we can guarantee that the allocation is EFX for IDO instances.

Lemma 3.1. *For any IDO instance, Algorithm 1 returns an EFX allocation in* $O(n^3m)$ *time.*

We prove Lemma 3.1 in Appendix. We remark that a parallel work [29] also proved that the envy-cycle elimination algorithm achieves EFX for IDO instances in the case of goods as well. Combining Lemmas 3.1 and 2.1, we have that for IDO instances, PROPX allocations can be computed in polynomial time by Algorithm 1.

The running time of Algorithm 1 is $O(n^3m)$, which is not comparable with that by Moulin [54] in terms of time complexity. The advantage of Algorithm 1 is that the envy-cycle elimination algorithm is widely used in fair division which satisfies some good properties such as approximately MMS. It is proved in [19] that the envy-cycle elimination algorithm, where the cycles are resolved arbitrarily, guarantees 4/3-MMS for chores. Since Algorithm 1 only uses one particular way to resolve the cycles, it continues to ensure 4/3-MMS.

Lemma 3.2 ([19]). *Algorithm 1 outputs a 4/3-MMS allocation.*

We complement this result with an example implying that the analysis of Lemma 3.2 is tight. Consider the following instance with n agents and $2n+1$ items where all agents have the same cost function shown in Table 1 and $\epsilon = 1/6(n-1)$. It can be verified that $\mathsf{MMS}_i = 1+2\epsilon$ for every agent i, and the corresponding allocation $\mathbf{X}^*$ has $X^*_j = \{e_j, e_{2n-j-1}\}$ for all $j \leq n-1$, and $X^*_n = \{e_{2n-1}, e_{2n}, e_{2n+1}\}$. Note that

$$c_i(X^*_j) = 1+2\epsilon \quad \text{and} \quad c_i(X^*_n) = 1+\epsilon.$$

However, the allocation returned by Algorithm 1 in the first $2n$ steps is $X_i = \{e_i, e_{2n-i+1}\}$ for all $i \in N$ and accordingly $c_i(X_i) = 1$. In the last step, no matter which agent obtains item e_{2n+1}, her cost will be 4/3. Thus the allocation is $4/(3(1+2\epsilon))$-MMS.

Item	e_1	$\cdots$	e_j	$\cdots$	e_n	e_{n+1}	$\cdots$	e_{n+j}	$\cdots$	e_{2n}	e_{2n+1}
Cost	$\frac{2}{3}$	$\cdots$	$\frac{2}{3}-(j-1)\epsilon$	$\cdots$	$\frac{1}{2}$	$\frac{1}{2}$	$\cdots$	$\frac{1}{2}-(j-1)\epsilon$	$\cdots$	$\frac{1}{3}$	$\frac{1}{3}$

Table 1: A Tight Example for Lemma 3.2

We conclude the above discussion by the following theorem, where the reduction to general instances will be proved in Lemma 4.1.

Theorem 3.1. *There is an algorithm that given any unweighted instance returns an allocation that is simultaneously PROPX and 4/3-MMS. In addition, if the instance is IDO, the allocation is EFX.*

4. Weighted PROPX Allocations

In this section, we focus on the general case where the agents may have different shares for the items. Specifically, each agent $i \in N$ has share $s_i > 0$, and $\sum_{i \in N} s_i = 1$. Intuitively, s_i represents how large a fraction of the chores should be completed by agent i.

4.1. Weighted PROPX Allocations

We first adapt our solution concepts to the weighted setting. The weighted proportionality of each agent is $\mathsf{WPROP}_i = s_i \cdot c_i(M)$. When the cost functions are normalized, $\mathsf{WPROP}_i = s_i$.

Definition 4.1 (WPROP and WPROPX)**.** *For any $\alpha \geq 1$, an allocation $\mathbf{X}$ is α-approximate weighted proportional (α-WPROP) if $c_i(X_i) \leq \alpha \cdot \mathsf{WPROP}_i$ for all $i \in N$. An allocation $\mathbf{X}$ is α-approximate weighted proportional up to any item (α-WPROPX) if $c_i(X_i \setminus \{e\}) \leq \alpha \cdot \mathsf{WPROP}_i$ for any agent $i \in N$ and any item $e \in X_i$. When $\alpha = 1$, allocation $\mathbf{X}$ is WPROP or WPROPX, respectively.*

As we have mentioned in Section 3, to design algorithms to compute PROPX and WPROPX allocations, it is without loss of generality to focus on the IDO instances. We present the lemma as follows and leave the formal proof in Appendix.

Lemma 4.1. *If there exists a polynomial time algorithm $\mathcal{A}$ that given any IDO instance computes an α-WPROPX allocation, then there exists a polynomial time algorithm $\mathcal{A}'$, whose running time is that of $\mathcal{A}$ plus $O(mn \log m)$, that given any instance computes an α-WPROPX allocation.*

Lemma 4.1 is the same as the counterpart reductions in [27, 19, 48] which are designed for (unweighted) MMS. It deserves to note that the reduction does not need to access the cardinal costs and thus holds for the ordinal setting as well.

Next, we show that any WPROPX allocation achieves a 2-approximation of AnyPrice share (APS) fairness [13]. We first adapt the APS fairness defined in [13] for goods to chores. The high-level idea is as follows. Each agent's weight is regarded as her loan from the system and she can obtain a reward by completing a chore to repay the loan. The agent's AnyPrice Share is then defined as the smallest cost she can guarantee by completing a set of

chores that suffices to repay the loan when the items' rewards are adversarially set with a total reward of 1. Let $\mathcal{R} = \{(r_1, \ldots, r_m) \mid r_j \geq 0 \text{ for all } e_j \in M \text{ and } \sum_{e_j \in M} r_j = 1\}$ be the set of item-reward vectors.

Definition 4.2 (AnyPrice Share)**.** *The AnyPrice Share (APS) of agent i with weight s_i is defined as*

$$\mathsf{APS}_i = \max_{(r_1,\ldots,r_m)\in\mathcal{R}} \min_{S\subseteq M} \left\{ c_i(S) \mid \sum_{e_j \in S} r_j \geq s_i \right\}.$$

For any $\alpha \geq 1$, an allocation $\mathbf{X}$ is α-approximate AnyPrice Share fair (α-APS) if $c_i(X_i) \leq \alpha \cdot \mathsf{APS}_i$ for any agent $i \in N$. When $\alpha = 1$, allocation $\mathbf{X}$ is APS fair.

Similar to Lemma 2.2, we have the following property regarding WPROPX and APS, which immediately implies that any algorithm that computes WPROPX allocations is 2-APS fair.

Lemma 4.2. *Any WPROPX allocation is 2-APS.*

Proof. Before proving the lemma, we first claim the following inequality for APS.

$$\mathsf{APS}_i \geq \max\{s_i, \max_{e\in M}\{c_i(e)\}\}. \tag{3}$$

To show $\mathsf{APS}_i \geq s_i$, it suffices to find a reward vector $(r_1, \ldots, r_m)$ such that every set S of items with reward no less than s_i is at least of cost s_i. Thus, we can simply set $r_j = c_i(e_j)$. Similarly, to show $\mathsf{APS}_i \geq c_i(e_{j^*})$, where $e_{j^*} = \arg\max_{e_j \in M}\{c_i(e_j)\}$, we set $r_{j^*} = 1$ and $r_j = 0$ for $j \neq j^*$. Then the unique way to repay the loan is to complete chore e_{j^*}, which incurs cost $c_i(e_{j^*})$. For any WPROPX allocation $\mathbf{X}$ and any agent i, we have $c_i(X_i \backslash \{e\}) \leq s_i$ for any $e \in X_i$. Thus $c_i(X_i) \leq s_i + \max_{e\in M}\{c_i(e)\} \leq 2 \cdot \mathsf{APS}_i$, where the last inequality follows from Inequality (3). □

However, an APS allocation can be $\Theta(n)$-WPROPX. Recall the example in Lemma 2.3. In that example, there are n agents and $m = n$ items, and for all agent $i \in N$ we have $c_i(e_1) = n - 1$ and $c_i(e_j) = 1$ for $j = 2, \ldots, n$. Thus

$$\mathsf{APS}_i \geq \max_{e\in M}\{c_i(e)\} = n - 1 \quad \text{and} \quad \mathsf{PROP}_i = \frac{2(n-1)}{n}.$$

Thus allocating $\{e_2, \ldots, e_n\}$ to some agent i is APS to her but has $\Theta(n)$ approximation regarding PROPX.

4.2. Bid-and-Take Algorithm

Note that the top-trading envy cycle elimination algorithm is not able to compute a WPROPX allocation for the weighted setting. Instead, in this section, we present the *bid-and-take* algorithm. In our algorithm, the items are allocated from the highest to the lowest cost. Moreover, each item is allocated to an *active* agent that minimizes the current social cost, i.e., who has minimum cost on the item among all active agents. Initially, all agents are active. When the cumulative cost of an agent exceeds her proportional share, we inactivate her.

Algorithm 2: Bid-and-Take

Input: IDO instance with $c_i(e_1) \geq \cdots \geq c_i(e_m)$ and $c_i(M) = 1$, $\forall i \in N$, and shares of agents $0 < s_1 \leq \cdots \leq s_n$ and $\sum_{i \in N} s_i = 1$.
Let $X_i \leftarrow \emptyset$, $\forall i \in N$ and $A \leftarrow N$ be the set of active agents.
for $j = 1, 2, \ldots, m$ **do**
    Let $i \in \arg\min_{i' \in A}\{c_{i'}(e_j)\}$, and set $X_i \leftarrow X_i \cup \{e_j\}$.
    if $c_i(X_i) > s_i$ **then**
        $A \leftarrow A \setminus \{i\}$.
Output: Allocation $\mathbf{X} = (X_1, \ldots, X_n)$.

The following example demonstrates the execution of Algorithm 2, and we can also observe that the returned allocation may not be PO either.

Example 4.1. *Consider an instance with two agents and six items, where the two agents have the same weight. The costs are shown in the following table.*

	$c_i(e_1)$	$c_i(e_2)$	$c_i(e_3)$	$c_i(e_4)$	$c_i(e_5)$	$c_i(e_6)$
1	*0.26*	*0.26*	*0.12*	*0.12*	*0.12*	*0.12*
2	*0.27*	*0.27*	*0.22*	*0.22*	*0.01*	*0.01*

The allocation $\mathbf{X}$ *returned by Algorithm 2 is shown in the squares. However, the two agents can exchange item* e_2 *and items* $\{e_3, e_4\}$ *to get an allocation* $\mathbf{X}'$*, where* $c_1(X_1') = c_1(\{e_1, e_3, e_4\}) = 0.5 < 0.52 = c_1(X_1)$ *and* $c_2(X_2') = c_2(\{e_2, e_5, e_6\}) = 0.29 < 0.44 = c_2(X_2)$*. Hence* $\mathbf{X}'$ *Pareto improves* $\mathbf{X}$*.*

Next, we prove the following property for Algorithm 2.

Claim 4.1. *At any point of Algorithm 2, for any active agents $i, i' \in A$,*

$$c_i(X_{i'}) \geq c_{i'}(X_{i'}).$$

Proof. Since we allocate each item to the active agent that has the smallest cost on the item, we have $c_i(e) \geq c_{i'}(e)$ for each item $e \in X_{i'}$ (because both agents i and i' are active when the item is allocated). Hence we have $c_i(X_{i'}) \geq c_{i'}(X_{i'})$, and Claim 4.1 holds. □

Lemma 4.3. *Algorithm 2 returns a WPROPX allocation in $O(mn)$ time for IDO instances.*

Proof. Note that in Algorithm 2, an agent becomes inactive as soon as $c_i(X_i) > s_i$, and no additional item will be allocated to this agent. Hence to show that the final allocation $\mathbf{X} = (X_1, \ldots, X_n)$ is WPROPX, it suffices to show that the algorithm allocates all items, i.e., the set of active agents A is non-empty when each item $j \in M$ is considered. Note that if all items are allocated, for any agent $i \in N$, we have $c_i(X_i \setminus \{e\}) \leq s_i$ for any item $e \in X_i$ by the fact that items are allocated in decreasing order of the cost.

Next, we show that $A \neq \emptyset$ when considering each item $e_j \in M$. Suppose when we consider some item e_j, all agents are inactive, i.e., $A = \emptyset$. Let i be the last agent that becomes inactive. At the moment when i becomes inactive, we have

$$c_i(M) \geq \sum_{i' \in N} c_i(X_{i'}) \geq \sum_{i' \in N} c_{i'}(X_{i'}) > \sum_{i' \in N} s_{i'} = 1,$$

where the second inequality follows by Claim 4.1 and the last inequality by the design of the algorithm, which is a contradiction with $c_i(M) = 1$.

The algorithm runs in $O(mn)$ time as there are m rounds and in each round, we only need to find an agent who has the smallest cost on this item. □

Combining Lemmas 4.3 and 4.1, we have the following theorem.

Theorem 4.1. *There is an algorithm that computes a WPROPX allocation for any weighted instance in polynomial time.*

5. Ordinal Setting

In this section, we investigate the extent to which we can compute approximately (weighted) PROPX allocations with ordinal preferences. Note that the problem becomes trivial if $m \leq n$, because as long as every agent gets at most one item, the allocation is PROPX. Thus in the following, we assume $m > n$. By Lemma 4.1, it suffices to consider the IDO instances, and thus we omit the additional running time of $O(mn \log m)$ for the reduction for simplicity when we analyze the running time of our algorithms.

5.1. Unweighted Setting

To highlight the intuition, we first consider the unweighted case, and present Algorithm 3 which always computes a 2-PROPX allocation in polynomial time. In the algorithm, we partition the agents into two groups: $N_1 = \{1, 2, \ldots, \lfloor n/2 \rfloor\}$ and $N_2 = N \setminus N_1$. We first allocate each agent in N_1 a large item and then run the round-robin algorithm where agents in N_2 take turns to select an item with the largest cost from the remaining items.

Algorithm 3: Ordinal Approximate PROPX Allocation

Input: IDO instance with $c_i(e_1) \geq \cdots \geq c_i(e_m)$ for all $i \in N$.
Initialize: $\mathbf{X} = (X_1, \ldots, X_n)$ where $X_i \leftarrow \emptyset$ for all $i \in N$.
for $j = 1, 2, \ldots, \lfloor \frac{n}{2} \rfloor$ **do**
    $X_j \leftarrow \{e_j\}$.
Let $i \leftarrow 1$.
for $j = \lfloor \frac{n}{2} \rfloor + 1, \lfloor \frac{n}{2} \rfloor + 2, \ldots, m$ **do**
    $X_{i+\lfloor \frac{n}{2} \rfloor} \leftarrow X_{i+\lfloor \frac{n}{2} \rfloor} \cup \{e_j\}$.
    $i \leftarrow (i \mod (\lfloor \frac{n}{2} \rfloor + 1)) + 1)$.
Output: Allocation $\mathbf{X} = (X_1, \ldots, X_n)$.

Theorem 5.1. *Algorithm 3 returns a 2-PROPX allocation in $O(m)$ time.*

Proof. It is straightforward that the algorithm runs in $O(m)$ time as the owner of each item is prefixed. Recall that $N_1 = \{1, 2, \ldots, \lfloor n/2 \rfloor\}$ and $N_2 = N \setminus N_1$. Thus $|N_1| = |N_2|$ if n is even and $|N_1| + 1 = |N_2|$ otherwise. Let $(X_1, \ldots, X_n)$ be the returned allocation. It is obvious that the allocation is

PROPX for all $i \in N_1$ as $|X_i| = 1$. Consider any agent $i \in N_2$. Denote by $X_i = \{e_1, \ldots, e_k\}$ the items allocated to i, where

$$c_i(e_1) \geq c_i(e_2) \geq \cdots \geq c_i(e_k).$$

Since the items allocated to agents in N_1 are those with largest costs, we have

$$c_i(e_1) \leq c_i(X_l) \quad \text{for all } l \in N_1. \tag{4}$$

Moreover, as the items $\{e_{\lfloor n/2 \rfloor + 1}, \ldots, e_m\}$ are allocated from the most costly to the least costly in a round-robin manner, we have

$$c_i(X_i \setminus \{e_1\}) \leq c_i(X_j) \quad \text{for all } j \in N_2 \setminus \{i\}.$$

Thus we have

$$\begin{aligned} |N_2| \cdot c_i(X_i \setminus \{e_1\}) &\leq c_i(X_i \setminus \{e_1\}) + \sum_{j \in N_2 \setminus \{i\}} c_i(X_j) \\ = \sum_{j \in N_2} c_i(X_j) - c_i(e_1) &= 1 - \sum_{l \in N_1} c_i(X_l) - c_i(e_1) \leq \ 1 - (|N_1| + 1) \cdot c_i(e_1), \end{aligned}$$

where the last inequality follows from Inequality (4). Thus

$$\begin{aligned} c_i(X_i) = c_i(e_1) + c_i(X_i \setminus \{e_1\}) &\leq c_i(e_1) + \frac{1}{|N_2|}\big(1 - (|N_1| + 1) \cdot c_i(e_1)\big) \\ = \frac{1}{|N_2|} + \left(1 - \frac{|N_1| + 1}{|N_2|}\right) \cdot c_i(e_1) &\leq \frac{1}{|N_2|} \leq \frac{2}{n} = 2 \cdot \mathsf{PROP}_i, \end{aligned}$$

where the second inequality follows from $|N_1| + 1 \geq |N_2|$, and the last inequality holds because $|N_2| \geq n/2$. □

Actually, regarding PROPX, the approximation ratio 2 our algorithm achieves is the best possible for ordinal algorithms, proved in the following lemma.

Lemma 5.1. *With only ordinal preferences, no algorithm can guarantee a better-than-2 approximation for PROPX.*

Proof. Consider an IDO instance with 2 agents and m items, where m is sufficiently large and $c_i(e_1) \geq \cdots \geq c_i(e_m)$ for both $i \in \{1, 2\}$. Without loss of generality, suppose item e_1 is allocated to agent 1. If $|X_1| > 1$, consider the cardinal costs for agent 1, $c_1(e_1) = 1$ and $c_1(e_j) = 0$ for all $j > 1$, and thus $c_1(X_1 \setminus \{e\}) = 2 \cdot \mathsf{PROP}_1$ for any $e \in X_1 \setminus \{e_1\}$. If $|X_1| = 1$, consider the cardinal costs for agent 2, $c_2(e) = 1/m$ for all $e \in M$, and thus $c_2(X_2 \setminus \{e\}) = 1 - 2/m \approx 2 \cdot \mathsf{PROP}_2$ for any $e \in X_2$. □

5.2. Weighted Setting

We next extend Algorithm 3 to the weighted setting. Given an arbitrary weighted instance with $s_1 \leq \cdots \leq s_n$, let

$$i^* = \max\{i \mid \textstyle\sum_{j=1}^{i} s_j \leq \frac{1}{2}\}.$$

We partition the agents into two groups: $N_1 = [i^*]$ and $N_2 = N \setminus N_1$. Let $w_1 = \sum_{i \in N_1} s_i$. It is not hard to see the following properties.

- By definition we have $w_1 \leq 1/2$.
- We have $i^* \geq n/2$ because $\sum_{i=1}^{n/2} s_i \leq 1/2$.
- We have $s_i > 1/(2i^* + 2)$ for all $i \in N_2$ because otherwise $s_{i^*+1} \leq 1/(2i^* + 2)$, which implies $\sum_{j=1}^{i^*+1} s_j \leq 1/2$. In other words, $i^* + 1$ should be included in N_1 as well, which is a contradiction.

As before, we assign each agent $j \in N_1$ a single item $e_j \in M$. Recall that these are the i^* items with the maximum costs. Let $M_L = \{e_1, e_2, \ldots, e_{i^*}\}$ be these items, and $M_S = M \setminus M_L$ be the remaining items. We call M_L the *large* items and M_S the *small* items. Then we run a weighted version of round robin algorithm by repeatedly allocating an item to the agent $i \in N_2$ with the minimum $|X_i|/s_i$ until all items are allocated (see Algorithm 4). The weighted round robin algorithm is proved to ensure weighted EF1 for indivisible goods in [32]. In Lemma 5.2, we prove that the weighted round robin algorithm also ensures weighted EF1 for IDO instances with chores. Recently, our approach was used in [58] to show the existence of weighted EF1 allocations for non-IDO instances.

Our main result relies on the following technical lemma.

Lemma 5.2. *For every agent $j \neq i$, we have*

$$\frac{c_i(X_i \setminus \{e_i\})}{s_i} \leq \frac{c_i(X_j)}{s_j}.$$

Proof. Suppose $X_i = \{o_1, o_2, \ldots, o_k\}$, where $c_i(o_1) \geq \cdots \geq c_i(o_k)$. Recall that $o_1 = e_i$. As we need to compared the costs of bundle $X_i \setminus \{e_i\}$ and X_j that are scaled by different weights, for convenience we translate the scaled cost into an integration as follows.

Algorithm 4: Ordinal Approximate WPROPX Allocation

Input: IDO instance, and the shares of agents $s_1 \le \cdots \le s_n$.
Initialize: $\mathbf{X} = (X_1, \ldots, X_n)$ where $X_i \leftarrow \emptyset$ for all $i \in N$.
Let $i^* = \max\{i \mid \sum_{j=1}^{i} s_j \le 1/2\}$, $N_1 = \{1, \ldots, i^*\}$ and $N_2 = N \setminus N_1$.
for $j = 1, 2, \ldots, i^*$ **do**
    $X_j \leftarrow \{e_j\}$.
for $j = i^* + 1, \ldots, m$ **do**
    Let $l \in \arg\min_l\{|X_l|/s_l\}$ where tie is broken by agent ID.
    $X_l \leftarrow X_l \cup \{e_j\}$.
Output: Allocation $\mathbf{X} = (X_1, \ldots, X_n)$.

We define a real-valued function $\rho : (0, k/s_i] \to \mathbb{R}^+$ such that:

$$\rho(\alpha) = c_i(o_t), \quad \text{for} \quad \frac{t-1}{s_i} < \alpha \le \frac{t}{s_i} \text{ and } t \in [k].$$

Thus, for all $t \in [k]$, we have

$$\frac{c_i(o_t)}{s_i} = \int_{\frac{t-1}{s_i}}^{\frac{t}{s_i}} \rho(\alpha)\, d\alpha \quad \text{and} \quad \frac{c_i(X_i \setminus \{e_i\})}{s_i} = \int_{\frac{1}{s_i}}^{\frac{k}{s_i}} \rho(\alpha)\, d\alpha.$$

On a high level, we can interpret the inclusion of item o_t to X_i as agent i continuously eats item o_t. Such process increases the scaled cost of agent i by $\frac{c_i(o_t)}{s_i}$ and increases $\frac{|X_i|}{s_i}$ from $\frac{t-1}{s_i}$ to $\frac{t}{s_i}$.

Similarly, we assume $X_j = \{o'_1, \ldots, o'_{k'}\}$, where $c_i(o'_1) \ge \cdots \ge c_i(o'_{k'})$, and define

$$\rho'(\alpha) = c_i(o'_t), \quad \text{for} \quad \frac{t-1}{s_j} < \alpha \le \frac{t}{s_j}.$$

By definition we have

$$\frac{c_i(X_j)}{s_j} = \int_{0}^{\frac{k'}{s_j}} \rho'(\alpha)\, d\alpha.$$

Recall that in Algorithm 4, each item is allocated to the agent $i \in N_2$ with the minimum $|X_i|/s_i$. Thus we have

$$\frac{k-1}{s_i} = \frac{|X_i| - 1}{s_i} \le \frac{|X_j|}{s_j} = \frac{k'}{s_j},$$

where the inequality holds because otherwise item o_k will not be allocated to agent i in Algorithm 4.

Next, we show that $\rho(\alpha) \le \rho'(\alpha - \frac{1}{s_i})$. Consider the round when $|X_i|/s_i$ reaches α. Suppose item o_t is allocated to agent i in this round, i.e., $\rho(\alpha) = c_i(o_t)$. Note that in this round, we must have $|X_j|/s_j \ge \alpha - 1/s_i$. Because otherwise when item o_t is considered we have $|X_j|/s_j < \alpha - 1/s_i \le |X_i|/s_i$, which means that item o_t should not be allocated to agent i. In other words, the event "$|X_j|/s_j$ reaches $\alpha - 1/s_i$" happens before the event "$|X_i|/s_i$ reaches α". Since items are allocated from the most costly to the least costly, we have $\rho(\alpha) = c_i(o_t) \le \rho'(\alpha - 1/s_i)$.

Combining the above discussion, we have

$$\begin{aligned}\frac{c_i(X_i \setminus \{e_i\})}{s_i} &= \int_{\frac{1}{s_i}}^{\frac{k}{s_i}} \rho(\alpha)\, d\alpha \le \int_{\frac{1}{s_i}}^{\frac{k}{s_i}} \rho'(\alpha - \frac{1}{s_i})\, d\alpha \\ &= \int_{0}^{\frac{k-1}{s_i}} \rho'(\alpha)\, d\alpha \le \int_{0}^{\frac{k'}{s_j}} \rho'(\alpha)\, d\alpha = \frac{c_i(X_j)}{s_j},\end{aligned}$$

which proves the lemma. □

Given the above lemma, we can obtain the following main result.

Theorem 5.2. *Algorithm 4 computes a 2-WPROPX allocation in $O(mn)$ time for any given weighted instance.*

Proof. It is straightforward that the algorithm runs in $O(mn)$ time as there are $O(m)$ rounds for the second for-loop and each round requires $O(n)$ time.

As before, it suffices to show that the allocation is 2-WPROP for N_2, i.e., $c_i(X_i) \le 2 \cdot s_i$ for all $i \in N_2$. In Algorithm 4, each agent $i \in N_2$ receives item $i \in M_S$ as her first item. Next we upper bound the total cost agent i receives excluding item i. By Lemma 5.2, we have

$$\sum_{j \in N_2} \frac{s_j}{s_i} \cdot c_i(X_i \setminus \{e_i\}) \le c_i(X_i \setminus \{e_i\}) + \sum_{j \in N_2 \setminus \{e_i\}} c_i(X_j) = c_i(M_S) - c_i(e_i).$$

Reordering the above inequality, we have

$$\begin{aligned}c_i(X_i \setminus \{e_i\}) &\le \frac{s_i}{\sum_{j \in N_2} s_j} \cdot (c_i(M_S) - c_i(e_i)) \\ &\le 2 \cdot s_i \cdot (c_i(M_S) - c_i(e_i)),\end{aligned}$$

where the second inequality holds because $\sum_{j \in N_2} s_j \geq 1/2$.

Since every item in M_L has cost at least $c_i(e_i)$ under the cost function of agent i, we have $c_i(M_L) \geq i^* \cdot c_i(e_i)$. Therefore we have

$$\begin{aligned} c_i(X_i) &= c_i(X_i \setminus \{e_i\}) + c_i(e_i) \\ &\leq 2 \cdot s_i \cdot (c_i(M_S) - c_i(e_i)) + \frac{c_i(M_L) + c_i(e_i)}{i^* + 1} \\ &< 2 \cdot s_i \cdot (c_i(M_S) - c_i(e_i)) + 2 \cdot s_i \cdot (c_i(M_L) + c_i(e_i)) \leq 2 \cdot s_i, \end{aligned}$$

where the second inequality holds because $s_i > 1/2(i^* + 1)$ and $c_i(M_L) + c_i(e_i) > 0$. □

6. Pareto Optimal and PROPX Allocations

We first prove that PROPX and PO are not compatible with any $n \geq 2$ agents and additive valuations, even when the agents have symmetric weights.

Theorem 6.1. *PROPX and PO are not compatible with any $n \geq 2$ agents and additive valuations, even when the agents have symmetric weights.*

Proof. Consider the following instance with $n \geq 2$ symmetric agents and $m = n + 1$ items. Let $0 < \epsilon < \frac{1}{n^2}$ be a small constant.

	$c_i(e_1)$	$c_i(e_2)$	$\dots$	$c_i(e_n)$	$c_i(e_{n+1})$
1	$\mathbf{0}$	ϵ	$\dots$	ϵ	$1-(n-1)\epsilon$
2	ϵ	$\mathbf{0}$	$\dots$	ϵ	$1-(n-1)\epsilon$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
n	ϵ	ϵ	$\dots$	$\mathbf{0}$	$1-(n-1)\epsilon$

Note that in this instance, any PO allocation must allocate item i to agent i, for all $i = 1, \cdots, n$. Otherwise reallocating item i from its owner to agent i does not hurt anyone but strictly decreases the original owner's cost. However, in such allocations, the agent who receives item $n + 1$ does not satisfy PROPX, since by removing the item with cost 0, the remaining cost is $1 - (n-1)\epsilon > \frac{1}{n}$. □

The hard example in the above proof is due to some items that are zero-valued by some agents. Thus, in the following of this section, we focus on the case when all items are positively valued by all agents:

Assumption ($\star$) $c_i(e) \neq 0$ for all $i \in N$ and $e \in M$.

We find that, even with Assumption ($\star$), PROPX and fPO are still not compatible, in contrast with [11] where an fPO and PROP1 allocation exists for chores.

Proposition 6.1. *There may not exist any PROPX and fPO allocation even with Assumption ($\star$) and for 2 agents with symmetric weights.*

Proof. Consider an instance with 2 agents and 3 items. The costs are shown as follows.

	$c_i(e_1)$	$c_i(e_2)$	$c_i(e_3)$
1	0.1	0.7	0.2
2	0.2	0.7	0.1

In the above instance, it is straightforward that any PROPX allocation cannot allocate all items to a single agent. By the characterization of fPO allocations for 2 agents [8], there are two fPO integral allocations, both of which allocate item e_1 to agent 1 and item e_3 to agent 2. However, neither of the two allocations is PROPX because the agent who receives item e_2 has cost 0.7 after removing the item with minimum cost. □

A similar result of Proposition 6.1 also appears in [41]. The proof of Proposition 6.1 shows that no rounding of an fPO and PROPX fractional allocation ensures PROPX. In the next two subsections, we identify several cases in which PROPX and PO allocations exist under Assumption ($\star$).

6.1. Two Agents

For the case of two agents, we show that PROPX and PO allocations are guaranteed to exist if they have symmetric weights. Actually, we prove a stronger statement that EFX and PO allocations exist for two symmetric agents. Since PROPX is implied by EFX, the result follows.

Proposition 6.2. *For two symmetric agents, EFX and PO allocations exist under* Assumption ($\star$).

Proof. Recall that we normalize the cost functions of the two agents $\{1, 2\}$ so that $c_1(M) = c_2(M) = 1$. Let (X_1, X_2) be the leximax allocation. In other words, among all allocations that minimizes $\max\{c_1(X_1), c_2(X_2)\}$, (X_1, X_2) has the minimum value of $\min\{c_1(X_1), c_2(X_2)\}$. In the following, we show that (X_1, X_2) is EFX and PO.

The Pareto optimality follows straightforwardly from the fact that the allocation that Pareto improves (X_1, X_2) must have a higher lexicographical order, which contradicts with (X_1, X_2) being Leximax. Next, we argue that the allocation is EFX. If the allocation is PROP, e.g., both agents $i \in \{1, 2\}$ have $c_i(X_i) \leq 0.5$, then it is clearly EFX because $c_i(X_j) = 1 - c_i(X_i) \geq 0.5$ for $j \neq i$. Now suppose that the allocation is not PROP. In other words, there is an agent, say agent 1, that has $c_1(X_1) > 0.5$. It follows that $c_1(X_2) = 1 - c_1(X_1) < 0.5$. We claim that $c_2(X_2) < 0.5$.

Assume otherwise, i.e., $c_2(X_1) \leq 0.5$. Thus swapping bundles X_1, X_2 gives a PROP allocation, which has $\max\{c_1(X_2), c_2(X_1)\} \leq 0.5$ and contradicts with (X_1, X_2) being Leximax. Therefore agent 2 does not envy agent 1, and the allocation is EFX to agent 2.

Next, we prove that the allocation is also EFX to agent 1. That is for any $e \in X_1$ (recall that $c_1(e) > 0$), we have $c_1(X_1 \setminus \{e\}) \leq c_1(X_2)$. Suppose otherwise, e.g., $c_1(X_1 \setminus \{e\}) > c_1(X_2)$. Then we have $c_1(X_1) > c_1(X_2 \cup \{e\})$. In other words, both $X_1 \setminus \{e\}$ and $X_2 \cup \{e\}$ offer a smaller cost to agent 1, compared to X_1. Now suppose we let agent 2 pick her preferred bundle between $X_1 \setminus \{e\}$ and $X_2 \cup \{e\}$, and assign the remaining one to agent 1. Then the cost of agent 2 is at most 0.5 while the cost of agent 1 is strictly smaller than $c_1(X_1)$, which contradicts with (X_1, X_2) being the Leximax allocation. □

The above result immediately implies the following.

Corollary 6.1. *For two symmetric agents, PROPX and PO allocations exist.*

Proposition 6.2 complements that of [55], who showed that for positive utilities and symmetric weights, the leximax is EFX and PO. Since the computation of Leximax allocation is NP-hard, our result only shows the existence of PROPX and PO allocations for two symmetric agents. A natural open question is whether there exist polynomial-time algorithms for the computation of such allocations.

When agents have asymmetric weights, the definition of Leximax allocation in the above proof can be generalized to first minimize $\max\{c_1(X_1) - s_1, c_2(X_2) - s_2\}$ and then minimize $\min\{c_1(X_1) - s_1, c_2(X_2) - s_2\}$. Unfortunately, such an allocation (which is PO) might not be WPROPX due to the following instance.

Example 6.1. *Consider the following instance with two agents having asymmetric weights $s_1 = 0.7$ and $s_2 = 0.3$.*

	$c_i(e_1)$	$c_i(e_2)$	$c_i(e_3)$	$c_i(e_4)$
1	*0.79*	*0.18*	*0.01*	*0.02*
2	*0.58*	*0.39*	*0.02*	*0.01*

It can be verified that the only Leximax allocation has $X_1 = \{e_1, e_3\}$ *and* $X_2 = \{e_2, e_4\}$*. Unfortunately, this allocation is not WPROPX.*

One may also wonder whether the Leximax allocation is PROPX and PO for $n \geq 3$ agents. Unfortunately, via the following example, we show that even for three symmetric agents, no Leximax allocation is PROPX and PO.

Example 6.2. *Consider the following instance with three symmetric agents and four items.*

	$c_i(e_1)$	$c_i(e_2)$	$c_i(e_3)$	$c_i(e_4)$
1	*0.34*	*0.05*	*0.31*	*0.3*
2	*0.4*	*0.4*	*0.09*	*0.11*
3	*0.4*	*0.4*	*0.11*	*0.09*

To minimize the maximum cost, the Leximax allocation has to assign items e_1, e_2 *to agent* 1*. Thus the unique Leximax allocation assigns the items as we indicated using the rectangles. However, this allocation is not PROPX since*

$$c_1(X_1 \setminus \{e_2\}) = c_1(e_1) = 0.34 > 1/3.$$

6.2. Restricted Cost Functions

In this section, we show that for the cases when agents have *lexicographic* or *bi-valued* cost functions, PROPX and PO allocations always exist and can be computed efficiently.

Definition 6.1 (Lexicographic Cost)**.** *We say that cost function* c *is* lexicographic *if there is a partition* $(L_1, \ldots, L_k)$ *of the items* M *such that*

1. $\forall i \in [k]$ *and* $e, e' \in L_i$*, we have* $c(e) = c(e')$*, and*
2. $\forall i \in [k-1]$ *and* $e \in L_i$*, we have* $c(e) > c(\cup_{j=i+1}^{k} L_j)$*.*

Under lexicographic cost functions, we say that an agent prefers item e' to item e if there exist $i < j$ such that $e \in L_i$ and $e' \in L_j$. An agent with lexicographic cost function has a cost for each item that is higher than all more preferred items combined.

Definition 6.2 (Bi-valued Cost)**.** *We say that cost function c is* bi-valued *if there exist constants $\beta > \alpha > 0$ such that $c(e) \in \{\alpha, \beta\}$ for all items $e \in M$.*

Given an allocation $\mathbf{X}$, we can build a *trading graph* $G(\mathbf{X}) = (V(\mathbf{X}), E(\mathbf{X}))$. In $G(\mathbf{X})$, the set of vertices $V(\mathbf{X})$ contains one vertex per item in M. Furthermore, for any two vertices e and e', there is a directed edge from e to e' if $c_i(e') \leq c_i(e)$, where i is the agent who receives item e in $\mathbf{X}$. If $c_i(e') < c_i(e)$, the edge is called *strict*. We say that $G(\mathbf{X})$ admits a *Pareto trading cycle* C if there is a cycle C in $G(\mathbf{X})$ that contains at least one strict edge. We say that allocation $\mathbf{Y}$ is a result of resolving trading cycle C if for each edge $(e, e') \in C$ with $e \in X_i$, it holds that $Y_i = (X_i \setminus \{e\}) \cup \{e'\}$. Note that given an allocation, the construction of $G(\mathbf{X})$ and finding a cycle therein can be done in polynomial time.

Lemma 6.1. *For any allocation $\mathbf{X}$, $G(\mathbf{X})$ contains at most $O(m^2)$ edges, and after resolving a trading cycle, the number of edges in $G(\mathbf{X})$ strictly decreases.*

Proof. $G(\mathbf{X})$ contains at most $O(m^2)$ edges since there are in total m items corresponding to m vertices and each item has degree at most m. For any agent i and $e \in X_i$, let $\delta_i(e)$ be the out-degree of e in $G(\mathbf{X})$. Then the number of edges in $G(\mathbf{X})$ equals $\sum_{i \in N} \sum_{e \in X_i} \delta_i(e)$. For any agent i and item $e \in X_i$, if e is involved in the trading cycle, denote by $r(e)$ the item following e, i.e., $e \to r(e)$ is an edge in the trading cycle; if e is not involved, $r(e) = e$. Then $\delta_i(e) \geq \delta_i(r(e))$ since the items can only be exchanged with weakly better ones. Note that since the trading cycle involves a strict edge, there exists an agent i^* and an item $e^* \in X_{i^*}$ such that $\delta_{i^*}(e^*) > \delta_{i^*}(r(e^*))$. Thus after resolving the trading cycle, the number of edges (i.e., $\sum_{i \in N} \sum_{e \in X_i} \delta_i(r(e))$) is strictly decreased. □

Lemma 6.2. *Resolving a Pareto trading cycle in a (weighted) PROPX allocation preserves (weighted) PROPX.*

Proof. Let $\mathbf{X}$ be a (weighted) PROPX allocation and $\mathbf{Y}$ be the result after resolving one trading cycle in $G(X)$. Let the agents involved in the trading cycle C be N_C. Since $\mathbf{X}$ is PROPX, for any $e \in X_i$

$$c_i\left(X_i \setminus \{e\}\right) \leq c_i(M) \cdot s_i.$$

For any $i \in N_C$ where $Y_i = X_i \cup \{\alpha\} \setminus \{\beta\}$ and $\beta \in X_i$, considering $\alpha \in Y_i$ gives

$$c_i\left(Y_i \setminus \{\alpha\}\right) = c_i\left(X_i \setminus \{\beta\}\right) \leq c_i(M) \cdot s_i.$$

For any $e \in Y_i \backslash \{\alpha\}$, we have $e \in X_i$. Since $c_i(\alpha) \leq c_i(\beta)$,

$$\begin{aligned} c_i(Y_i \backslash \{e\}) &= c_i((X_i \cup \{\alpha\} \setminus \{\beta\}) \setminus \{e\}) \\ &= c_i(X_i \backslash \{e\}) + (c_i(\alpha) - c_i(\beta)) \\ &\leq c_i(X_i \backslash \{e\}). \end{aligned}$$

Hence

$$c_i(Y_i \backslash \{o\}) \leq c_i(X_i \backslash \{e\}) \leq c_i(M) \cdot s_i.$$

For any $i \notin N_C$ where $Y_i = X_i$, we have

$$c_i(Y_i) = c_i(X_i) \leq c_i(M) \cdot s_i.$$

Therefore, $\mathbf{Y}$ is (weighted) PROPX. □

For instances with lexicographic or bi-valued cost functions, the following lemma has been established for efficient checking of whether an allocation is PO or not. We remark that the following lemma is not true if the cost functions are general additive.

Lemma 6.3 ([7], [35])**.** *An allocation* $\mathbf{X}$ *is not PO with respect to lexicographic or bi-valued cost functions if and only if there exists a cycle in* $G(\mathbf{X})$ *which contains at least one edge corresponding to a strict preference.*

Utilizing these results, we show that there exist efficient algorithms for the computation of PROPX and PO allocations for instances when agents have lexicographic or bi-valued cost functions, even if they have asymmetric weights.

Proposition 6.3. *For lexicographic or bi-valued cost functions with* Assumption $(\star)$*, there exists a polynomial-time algorithm that computes allocations that are (weighted) PROPX and PO for agents with asymmetric weights.*

Proof. We only prove for lexicographic cost functions and the proof for bi-valued cost functions is the same. We first use any existing polynomial-time algorithm to compute a PROPX allocation for a group of agents with asymmetric weights. From Lemma 6.3, we can check in $O(nm)$ time whether the allocation is PO by constructing $G(\mathbf{X})$ and checking whether there is a cycle in $G(\mathbf{X})$ that contains at least one strict edge. If it is not, from Lemma 6.3, we know that the allocation's trading graph admits a Pareto trading cycle. If we resolve such a cycle, it follows from Lemma 6.2 that the new allocation is also (weighted) PROPX. By Lemma 6.1, there can be at most nm such Pareto improvements until the process terminates with an allocation that is both (weighted) PROPX and PO. □

7. Conclusion

In this paper, we studied the fair allocation of indivisible chores under the fairness notion of PROPX. We showed that PROPX allocations exist and can be computed efficiently for both symmetric and asymmetric agents. The returned allocations achieve the optimal guarantee on the price of fairness. We also designed the optimal approximation algorithms to compute (weighted) PROPX allocations with ordinal preferences. As byproducts, our results imply a 2-approximate algorithm for APS allocations for chores, and the existence of EFX and weighted EF1 allocations for IDO instances.

There are many future directions that are worth exploring. To name a few, as we have discussed, the existence or approximation of EFX is less explored for chores than for goods. Furthermore, we proved that any WPROPX allocation is 2-approximate APS, but it does not have a good guarantee for weighted MMS defined in [9]. It is still unknown whether weighted MMS admits constant approximations. Finally, we believe it is an important problem to investigate the compatibility between PROPX and Pareto optimality in the general setting when all items have positive cost to all agents. We discuss some challenges in Appendix D in this regard.

References

[1] H. Akrami and J. Garg. Breaking the 3/4 barrier for approximate maximin share. *CoRR*, abs/2307.07304, 2023.

[2] H. Akrami, J. Garg, and S. Taki. Improving approximation guarantees for maximin share. *arXiv preprint arXiv:2307.12916*, 2023.

[3] G. Amanatidis, H. Aziz, G. Birmpas, A. Filos-Ratsikas, B. Li, H. Moulin, A. A. Voudouris, and X. Wu. Fair division of indivisible goods: Recent progress and open questions. *Artif. Intell.*, 322:103965, 2023.

[4] G. Amanatidis, G. Birmpas, A. Filos-Ratsikas, A. Hollender, and A. A. Voudouris. Maximum nash welfare and other stories about EFX. *Theoretical Computer Science*, 863:69–85, 2021.

[5] G. Amanatidis, G. Birmpas, and E. Markakis. On truthful mechanisms for maximin share allocations. In *IJCAI*, pages 31–37. IJCAI/AAAI Press, 2016.

[6] G. Amanatidis, E. Markakis, and A. Ntokos. Multiple birds with one stone: Beating 1/2 for EFX and GMMS via envy cycle elimination. *Theor. Comput. Sci.*, 841:94–109, 2020.

[7] H. Aziz, P. Biro, J. Lang, J. Lesca, and J. Monnot. Efficient reallocation under additive and ordinal preferences. *Theoretical Computer Science*, 2019.

[8] H. Aziz, S. Brânzei, A. Filos-Ratsikas, and S. K. S. Frederiksen. The adjusted winner procedure: Characterizations and equilibria. In *IJCAI*, pages 454–460. AAAI Press, 2015.

[9] H. Aziz, H. Chan, and B. Li. Weighted maxmin fair share allocation of indivisible chores. In *IJCAI*, pages 46–52. ijcai.org, 2019.

[10] H. Aziz, B. Li, and X. Wu. Approximate and strategyproof maximin share allocation of chores with ordinal preferences. *Mathematical Programming*, 2022.

[11] H. Aziz, H. Moulin, and F. Sandomirskiy. A polynomial-time algorithm for computing a pareto optimal and almost proportional allocation. *Oper. Res. Lett.*, 48(5):573–578, 2020.

[12] H. Aziz, G. Rauchecker, G. Schryen, and T. Walsh. Algorithms for maxmin share fair allocation of indivisible chores. In *AAAI*, pages 335–341. AAAI Press, 2017.

[13] M. Babaioff, T. Ezra, and U. Feige. Fair-share allocations for agents with arbitrary entitlements. *EC*, 2021.

[14] M. Babaioff, N. Nisan, and I. Talgam-Cohen. Competitive equilibrium with indivisible goods and generic budgets. *Mathematics of Operations Research*, 46(1):382–403, 2021.

[15] A. Baklanov, P. Garimidi, V. Gkatzelis, and D. Schoepflin. Achieving proportionality up to the maximin item with indivisible goods. In *AAAI*, pages 5143–5150. AAAI Press, 2021.

[16] A. Baklanov, P. Garimidi, V. Gkatzelis, and D. Schoepflin. Propm allocations of indivisible goods to multiple agents. In *IJCAI*, pages 24–30. ijcai.org, 2021.

[17] S. Barman, U. Bhaskar, and N. Shah. Optimal bounds on the price of fairness for indivisible goods. In *WINE*, volume 12495 of *Lecture Notes in Computer Science*, pages 356–369. Springer, 2020.

[18] S. Barman, A. Biswas, S. K. K. Murthy, and Y. Narahari. Groupwise maximin fair allocation of indivisible goods. In *AAAI*, pages 917–924. AAAI Press, 2018.

[19] S. Barman and S. K. Krishnamurthy. Approximation algorithms for maximin fair division. In *EC*, pages 647–664. ACM, 2017.

[20] S. Barman and S. K. Krishnamurthy. On the proximity of markets with integral equilibria. In *AAAI*, pages 1748–1755. AAAI Press, 2019.

[21] S. Barman, S. K. Krishnamurthy, and R. Vaish. Finding fair and efficient allocations. In *EC*, pages 557–574. ACM, 2018.

[22] D. Baumeister, S. Bouveret, J. Lang, N. Nguyen, T. T. Nguyen, J. Rothe, and A. Saffidine. Positional scoring-based allocation of indivisible goods. *Autonomous Agents and Multi-Agent Systems*, 31(3):628–655, 2017.

[23] X. Bei, X. Lu, P. Manurangsi, and W. Suksompong. The price of fairness for indivisible goods. In *IJCAI*, pages 81–87. ijcai.org, 2019.

[24] B. Berger, A. Cohen, M. Feldman, and A. Fiat. (almost full) EFX exists for four agents (and beyond). *CoRR*, abs/2102.10654, 2021.

[25] D. Bertsimas, V. F. Farias, and N. Trichakis. The price of fairness. *Oper. Res.*, 59(1):17–31, 2011.

[26] U. Bhaskar, A. R. Sricharan, and R. 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.

[27] S. Bouveret and M. Lemaître. Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Auton. Agents Multi Agent Syst.*, 30(2):259–290, 2016.

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

[29] I. Caragiannis, J. Garg, N. Rathi, E. Sharma, and G. Varricchio. Existence and computation of epistemic efx allocations, 2022.

[30] I. Caragiannis, C. Kaklamanis, P. Kanellopoulos, and M. Kyropoulou. The efficiency of fair division. In *WINE*, volume 5929 of *Lecture Notes in Computer Science*, pages 475–482. Springer, 2009.

[31] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang. The unreasonable fairness of maximum nash welfare. *ACM Trans. Economics and Comput.*, 7(3):12:1–12:32, 2019.

[32] M. Chakraborty, A. Igarashi, W. Suksompong, and Y. Zick. Weighted envy-freeness in indivisible item allocation. In *AAMAS*, pages 231–239. International Foundation for Autonomous Agents and Multiagent Systems, 2020.

[33] B. R. Chaudhury, J. Garg, and K. Mehlhorn. EFX exists for three agents. In *EC*, pages 1–19. ACM, 2020.

[34] V. Conitzer, R. Freeman, and N. Shah. Fair public decision making. In *EC*, pages 629–646. ACM, 2017.

[35] S. Ebadian, D. Peters, and N. Shah. How to fairly allocate easy and difficult chores. In *AAMAS*, pages 372–380. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022.

[36] A. Farhadi, M. Ghodsi, M. T. Hajiaghayi, S. Lahaie, D. M. Pennock, M. Seddighin, S. Seddighin, and H. Yami. Fair allocation of indivisible goods to asymmetric agents. *J. Artif. Intell. Res.*, 64:1–20, 2019.

[37] U. Feige and X. Huang. On picking sequences for chores. In *EC*, pages 626–655. ACM, 2023.

[38] U. Feige, A. Sapir, and L. Tauber. A tight negative example for MMS fair allocations. In *WINE*, volume 13112 of *Lecture Notes in Computer Science*, pages 355–372. Springer, 2021.

[39] J. Garg and A. Murhekar. Computing pareto-optimal and almost envy-free allocations of indivisible goods. *CoRR*, abs/2204.14229, 2022.

[40] J. Garg, A. Murhekar, and J. Qin. Fair and efficient allocations of chores under bivalued preferences. In *AAAI*, pages 5043–5050. AAAI Press, 2022.

[41] J. Garg, A. Murhekar, and J. Qin. Improving fairness and efficiency guarantees for allocating indivisible chores. *CoRR*, abs/2212.02440, 2022.

[42] J. Garg and S. Taki. An improved approximation algorithm for maximin shares. In *EC*, pages 379–380. ACM, 2020.

[43] D. Halpern and N. Shah. Fair and efficient resource allocation with partial information. *CoRR*, abs/2105.10064, 2021.

[44] S. Heydrich and R. van Stee. Dividing connected chores fairly. *Theor. Comput. Sci.*, 593:51–61, 2015.

[45] F. Höhne and R. van Stee. Allocating contiguous blocks of indivisible chores fairly. *Information and Computation*, page 104739, 2021.

[46] H. Hosseini and K. Larson. Multiple assignment problems under lexicographic preferences. In *AAMAS*, pages 837–845. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[47] H. Hosseini, S. Sikdar, R. Vaish, and L. Xia. Fair and efficient allocations under lexicographic preferences. In *AAAI*, pages 5472–5480. AAAI Press, 2021.

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

[49] X. Huang and E. Segal-Halevi. A reduction from chores allocation to job scheduling. In *EC*, page 908. ACM, 2023.

[50] R. E. Korf. A complete anytime algorithm for number partitioning. *Artificial Intelligence*, 106(2):181–203, 1998.

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

[52] B. Li, Y. Li, and X. Wu. Almost (weighted) proportional allocations for indivisible chores. In *WWW*, pages 122–131. ACM, 2022.

[53] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi. On approximately fair allocations of indivisible goods. In *EC*, pages 125–131. ACM, 2004.

[54] H. Moulin. Fair division in the internet age. *Annual Review of Economics*, 11:1–37, 2019.

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

[56] J. Robertson and W. Webb. *Cake-cutting algorithms: Be fair if you can*. CRC Press, 1998.

[57] H. Steihaus. The problem of fair division. *Econometrica*, 16:101–104, 1948.

[58] X. Wu, C. Zhang, and S. Zhou. Weighted EF1 allocations for indivisible chores. In *EC*, page 1155. ACM, 2023.

Appendix A. Price of Fairness

In this section, we show that the allocation returned by Algorithm 2 achieves the optimal price of fairness (PoF) among all (weighted) PROPX allocations. Price of fairness is used to measure how much social welfare we lose if we want to maintain fairness among the agents. In this section, we always assume $c_i(M) = 1$ for all agents $i \in N$. Let $\Omega(\mathcal{I})$ be the set of all WPROPX allocations for instance $\mathcal{I}$, the price of fairness is defined as the worst-case ratio between the optimal (minimum) social cost $\text{opt}(\mathcal{I})$ without any constraints and the social cost under WPROPX allocations:

$$\mathsf{PoF} = \max_{\mathcal{I}} \min_{\mathbf{X} \in \Omega(\mathcal{I})} \frac{\text{sc}(\mathbf{X})}{\text{opt}(\mathcal{I})}.$$

Note that for any instance $\mathcal{I}$, $\text{opt}(\mathcal{I})$ is obtained by allocating every item to the agent who has smallest cost on it. Moreover, the assumption of $c_i(M) = 1$ for all agent $i \in N$ is necessary: if there exist two agents i and j having very different values of $c_i(M)$ and $c_j(M)$, then we have unbounded PoF even in the unweighted and IDO setting because the socially optimal allocation can allocate all items to one agent while WPROPX allocations cannot.

Lemma Appendix A.1. *Letting* $\mathbf{X}$ *be the allocation returned by Algorithm 2, we have* $\text{sc}(\mathbf{X}) \leq 1$.

Proof. Let $i \in N$ be the agent that receives the last item m. By Claim 4.1, for all agent $j \neq i$, we have $c_i(X_j) \geq c_j(X_j)$, because agent i is active throughout the whole allocation process. Hence we have $\text{sc}(\mathbf{X}) = \sum_{j \in N} c_j(X_j) \leq \sum_{j \in N} c_i(X_j) = c_i(M) = 1$. □

We show that Algorithm 2 achieves the optimal PoF.

Theorem Appendix A.1. *For the unweighted case, the PROPX allocation returned by Algorithm 2 achieves the optimal* PoF*, which is* $\Theta(n)$.

Proof. We first prove that the PoF is $\Omega(n)$ by giving an unweighted instance $\mathcal{I}$ for which any PROPX allocation $\mathbf{X}$ satisfies $\text{sc}(\mathbf{X}) \geq (n/6) \cdot \text{opt}(\mathcal{I})$. In $\mathcal{I}$, we have n agents and $m = n$ items with cost functions shown in the table below.

	$c_i(e_1)$	$\cdots$	$c_i(e_{n-1})$	$c_i(e_n)$
1	$2/n^2$	$\cdots$	$2/n^2$	$1-2(n-1)/n^2$
2	$1/n$	$\cdots$	$1/n$	$1/n$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
n	$1/n$	$\cdots$	$1/n$	$1/n$

For the above instance, we have

$$\mathrm{opt}(\mathcal{I}) = (n-1)\cdot\frac{2}{n^2}+\frac{1}{n} < \frac{3}{n}.$$

However, any PROPX allocation $\mathbf{X}$ allocates at most $n/2+1$ items to agent 1 because otherwise the bundle she receives has cost larger than $1/n$ even after removing one item. Hence we have

$$\mathrm{sc}(\mathbf{X}) \geq (\frac{n}{2}+1)\cdot\frac{2}{n^2}+(\frac{n}{2}-1)\cdot\frac{1}{n} > \frac{1}{2} \geq \frac{n}{6}\cdot\mathrm{opt}(\mathcal{I}).$$

Next, we show that for any unweighted instance $\mathcal{I}$, the allocation $\mathbf{X}$ computed by Algorithm 2 satisfies $\mathrm{sc}(\mathbf{X}) \leq n\cdot\mathrm{opt}(\mathcal{I})$. By Lemma Appendix A.1, it suffices to consider the case when $\mathrm{opt}(\mathcal{I}) < 1/n$. We show that in this case, we have $\mathrm{sc}(\mathbf{X}) = \mathrm{opt}(\mathcal{I})$. This is because, before any agent becomes inactive, we always allocate an item to the agent that has smallest cost on the item, as in the socially optimal allocation. Since $\mathrm{opt}(\mathcal{I}) < 1/n$, Algorithm 2 never turns any agent into inactive, which implies that $\mathrm{sc}(\mathbf{X}) = \mathrm{opt}(\mathcal{I})$. □

We note that the hard instance used to show Theorem Appendix A.1 is IDO, which means the PoF is $\Theta(n)$ even for the unweighted IDO instances. For the weighted case, we have the following result.

Theorem Appendix A.2. *For the weighted case, we have unbounded PoF. For IDO instances, Algorithm 2 computes a WPROPX allocation with optimal PoF, which is $\Theta(m)$.*

Proof. We first show that the PoF is unbounded for weighted non-IDO instances by giving the following hard instance $\mathcal{I}$ (with $s_1 = 1-\epsilon^2$ and $s_2 = \epsilon^2$) shown in the table below. It is easy to see that $\mathrm{opt}(\mathcal{I}) = 2\epsilon$. However, since any WPROPX allocation $\mathbf{X}$ allocates at most one item to agent 2, we have $\mathrm{sc}(\mathbf{X}) \geq 1/2$. Since $\epsilon > 0$ can be arbitrarily close to 0, we have an unbounded PoF for the weighted non-IDO instances.

	$c_i(e_1)$	$c_i(e_2)$	$c_i(e_3)$
1	0.5	0.5	0
2	ϵ	ϵ	$1-2\epsilon$

Next, we show that the PoF is $\Omega(m)$ for weighted IDO instances, by giving the following hard instance.

	$c_i(e_1)$	$c_i(e_2)$	$c_i(e_3)$	$\cdots$	$c_i(e_m)$
1	0.5	0.5	0	$\cdots$	0
2	$1/m$	$1/m$	$1/m$	$\cdots$	$1/m$

It is easy to see that $\text{opt}(\mathcal{I}) = 2/m$. However, for $s_1 = 1 - 1/m^2$ and $s_2 = 1/m^2$, since any WPROPX allocation $\mathbf{X}$ allocates at most one item to agent 2, we have $\text{sc}(\mathbf{X}) \geq 1/2 \geq m/4 \cdot \text{opt}(\mathcal{I})$.

Finally, we show that the weighted IDO instances the allocation $\mathbf{X}$ returned by Algorithm 2 satisfies $\text{sc}(\mathbf{X}) \leq m \cdot \text{opt}(\mathcal{I})$. By Lemma Appendix A.1, $\text{sc}(\mathbf{X}) \leq 1$. Moreover, for IDO instances,

$$\text{opt}(\mathcal{I}) = \sum_{e \in M} \min_{i \in N}\{c_i(e)\} \geq \min_{i \in N}\{c_i(e_1)\} \geq \frac{1}{m} \geq \frac{1}{m} \cdot \text{sc}(\mathbf{X}).$$

Hence allocation $\mathbf{X}$ achieves the asymptotically optimal PoF. □

Appendix B. Proof of Lemma 3.1

Proof. Envy cycle elimination algorithm runs in $O(n^3m)$ time [53, 26]. Roughly, by allocating a new item to an agent, at most n edges will be added to the graph, and by resolving a cycle, at least two edges will be removed. Therefore, there will be $O(mn)$ rounds of cycle resolution. In each round, finding a cycle needs $O(n^2)$ time, resulting in total running time $O(n^3m)$.

In the following, we prove by induction that the returned allocation is EFX. First, if no item is allocated to any agent, the allocation is trivially EFX. Let $\mathbf{X}$ be a partial and EFX allocation at the beginning of any round in Algorithm 1. Let X_0 be the set of all unassigned items. We prove that at the end of this round, the new partial allocation is also EFX. We show this by proving the following two claims.

Claim Appendix B.1. *Adding a new item to the allocation preserves EFX.*

Let i be any sink agent in G_X. By definition $c_i(X_i) \leq c_i(X_j)$ for all $j \in N$. Let e be the item with largest cost in X_0, which will be added to X_i. Since

the items are assigned from the most costly to least costly, and all agents have the same ordinal preference, $c_i(e') \geq c_i(e)$ for all $e' \in X_i$. Thus for any $e' \in X_i \cup \{e\}$ and any $j \neq i$,

$$c_i(X_i \cup \{e\} \setminus \{e'\}) \leq c_i(X_i \cup \{e\} \setminus \{e\}) = c_i(X_i) \leq c_i(X_j).$$

Thus Claim Appendix B.1 holds.

Claim Appendix B.2. *Resolving a top-trading envy cycle preserves EFX.*

Suppose we reallocate the bundles according to a top-envy cycle $C = (i_1, \ldots, i_d)$ in G_X. For any agent i who is not in the cycle, her bundle is not changed by the reallocation. Although other bundles are reallocated, the items in each bundle are not changed and thus the cycle-swapped allocation is still EFX for agent i. For any agent i in C, she will obtain her best bundle in this partial allocation $\mathbf{X}$, and hence the cycle-swapped allocation is EF for agent i. Thus Claim Appendix B.2 holds. Combining the two claims, at the end of each round, the partial allocation remains EFX. □

Appendix C. Proof of Lemma 4.1

Proof. In the following, we explicitly write $\mathcal{I} = (N, \mathbf{s}, M, \mathbf{c})$ to denote an instance with item set M, agent set N, weight vector $\mathbf{s} = (s_1, \ldots, s_n)$, and cost functions $\mathbf{c} = (c_1, \ldots, c_n)$. Given any instance $\mathcal{I} = (N, \mathbf{s}, M, \mathbf{c})$, we construct an IDO instance $\mathcal{I}' = (N, \mathbf{s}, M, \mathbf{c}')$ where $\mathbf{c}' = (c'_1, \ldots, c'_n)$ is defined as follows. Note that the construction of the IDO instance can be done in $O(mn \log m)$ time. Let $\sigma_i(j) \in M$ be the j-th most costly item under cost function c_i. Let $c'_i(e_j) = c_i(\sigma_i(j))$. Thus with cost functions $\mathbf{c}'$, the instance $\mathcal{I}'$ is IDO, in which all agents i has

$$c'_i(e_1) \geq c'_i(e_2) \geq \cdots \geq c'_i(e_m).$$

Then we run the algorithm for IDO instances on instance $\mathcal{I}'$, and get an α-WPROPX allocation $\mathbf{X}'$ for $\mathcal{I}'$. By definition, for all agents $i \in N$ we have

$$c'_i(X'_i \setminus \{e\}) \leq \alpha \cdot s_i, \qquad \forall e \in X'_i.$$

In the following, we use $\mathbf{X}'$ to guide us on computing a α-WPROPX allocation $\mathbf{X}$ for instance $\mathcal{I}$.

Recall that in the IDO instance $\mathcal{I}'$, for all agents, item e_1 has the maximum cost and item e_m has the minimum cost. We initialize $X_i = \emptyset$ for

all $i \in N$ and let $X_0 = M$ be the unallocated items. Sequentially for $j = m, m-1, \ldots, 1$, we let the agent i that receives item e_j under allocation $\mathbf{X}'$, i.e., $e_j \in X'_i$, pick her favourite unallocated item. Note that the order of items are well-defined in the IDO instance $\mathcal{I}'$. Specifically, we move item $e = \arg\min_{e' \in X_0}\{c_i(e')\}$ from X_0 to X_i. Thus we have $|X_i| = |X'_i|$ for each agent $i \in N$. Furthermore, we show that there is a bijection $f_i : X_i \to X'_i$ such that for any item $e \in X_i$, we have $c_i(e) \leq c'_i(e')$, where $e' = f_i(e)$. Recall that $c'_i(e_j) \geq c'_i(e_k)$ for all $k \geq j$. By the way c'_i is constructed, we know that there are at least $m - j + 1$ items that have cost at most $c'_i(e_j)$, under cost function c_i. Observe that when e is chosen from X_0, we have $|X_0| \geq j$. Hence there must exists an item e' in X_0 with cost $c_i(e') \leq c'_i(e_j)$. Since e has minimum cost among items in X_0 under cost function c_i, we have $c_i(e) \leq c'_i(e_j)$. Therefore, for any agent i and any $e \in X_i$, we have

$$c_i(X_i \setminus \{e\}) \leq c'_i(X'_i \setminus \{f_i(e)\}) \leq \alpha \cdot s_i,$$

where the last inequality follows because $\mathbf{X}'$ is α-WPROPX for instance $\mathcal{I}'$. Finally, it is easy to verify that the MMS benchmark in both instances are the same, while the cost of each agent in the general instance is smaller. Thus the algorithm guarantees the same approximation ratio for MMS. □

Appendix D. Challenges in Finding PROPX and PO Allocations

To find an allocation that is PROPX and PO, a naive approach would be to apply Pareto improvements on PROPX allocations. However, there are several challenges. First, checking whether a given allocation is PO is coNP-hard.

Proposition Appendix D.1. *Under additive cost functions, testing whether a given allocation is PO or not is weakly coNP-complete, even for $n = 2$ with identical weights.*

Proof. The proof is an adaptation of a similar result in [7] for positive utilities. First, given an allocation $\mathbf{X} = (X_1, \ldots, X_n)$, testing whether $\mathbf{X}$ is PO is in coNP since for any allocation $\mathbf{X}'$ one can test whether $\mathbf{X}$ is Pareto dominated by $\mathbf{X}'$ in linear time by comparing every agent's cost under the two allocations.

Next, we design a polynomial-time reduction from the PARTITION problem, which is a well-known NP-complete problem [50]. An instance of PARTITION is described by a set of t elements $E = \{e_1, \ldots, e_t\}$ where each

$e_j \in E$ has integer weight $w(e_j)$ such that $\sum_{e_i \in E} w(e_i) = 2M$. The question is to decide whether there is a balanced partition of E i.e., $S \subseteq E$ such that $\sum_{e_i \in S} w(e_i) = \sum_{e_i \in E \setminus S} w(e_i) = M$. Given any PARTITION instance, we construct a fair allocation instance with $t+1$ items $\{o^+, o_1, \ldots, o_t\}$ and two agents $\{1, 2\}$. Agent 1's cost function is: $c_1(o^+) = M$ and $c_1(o_i) = w(e_i)$ for all $i \in \{1, \ldots t\}$. Agent 2' cost function is: $c_2(o^+) = M + \varepsilon$, with $0 < \varepsilon < 1$, and $c_2(o_i) = w(e_i)$ for all $i \in \{1, \ldots, t\}$. Consider allocation $\mathbf{X}$ with $X_1 = \{o^+\}$ and $X_2 = O \setminus \{o^+\}$. Then $\mathbf{X}$ is PO if and only if there is a balanced partition of E. □

The second difficulty is that given an arbitrary PROPX allocation $\mathbf{X}$ that is not PO, any Pareto improvement of $\mathbf{X}$ makes $\mathbf{X}$ not PROPX, as shown by the following example.

Example Appendix D.1. *Consider an instance with four items and two agents with costs shown in the following table.*

	$c_i(e_1)$	$c_i(e_2)$	$c_i(e_3)$	$c_i(e_4)$
1	*0.32*	*0.27*	*0.11*	*0.3*
2	*0.13*	*0.05*	*0.38*	*0.44*

The allocation $\mathbf{X}$ *shown in squares, i.e.,* $X_1 = \{e_2, e_3\}$ *and* $X_2 = \{e_1, e_4\}$*, is PROPX but is not PO because it can be Pareto improved by allocation* $\mathbf{X}'$ *shown below, i.e.,* $X'_1 = \{e_4\}$ *and* $X'_2 = \{e_1, e_2, e_3\}$*. Allocation* $\mathbf{X}'$ *is not PROPX since* $c_2(X'_2 \setminus \{e_2\}) = 0.51 > 0.5$*.*

	$c_i(e_1)$	$c_i(e_2)$	$c_i(e_3)$	$c_i(e_4)$
1	*0.32*	*0.27*	*0.11*	*0.3*
2	*0.13*	*0.05*	*0.38*	*0.44*

Actually, $\mathbf{X}'$ *is the only allocation that Pareto improves* $\mathbf{X}$*. Suppose any allocation* $\mathbf{Y}$ *that Pareto dominates* $\mathbf{X}$*. Agent* 1*'s cost can be decreased by either giving one of her items to agent* 2 *without receiving any other item, or giving out both items* e_2 *and* e_3 *in exchange for either item* e_1 *or* e_4 *with a lower total cost. The first case is not acceptable since it strictly increases agent* 2*'s cost. For the second case, if agent* 1 *gets item* e_1*, then items* e_2*,* e_3 *and* e_4 *are assigned to agent* 2 *which increases her cost since*

$$c_2(Y_2) = c_2(\{e_2, e_3, e_4\}) = 0.87 > 0.57 = c_2(X_2).$$

Then the only option is that agent 1 *gets* e_4*, with*

$$c_1(Y_1) = c_1(e_4) = 0.3 < c_1(X_1), \text{ and}$$

$$c_2(Y_2) = c_2(\{e_1, e_2, e_3\}) = 0.56 < c_2(X_2).$$

Hence $\mathbf{X}'$ *is the only allocation that Pareto dominates* $\mathbf{X}$*.*

Therefore, for the allocation $\mathbf{X}$*, there does not exist a Pareto improvement over it that preserves PROPX.*