Almost (Weighted) Proportional Allocations for Indivisible Chores*†

Bo Li
Department of Computing
Hong Kong Polytechnic University
Hong Kong, China
comp-bo.li@polyu.edu.hk

Yingkai Li
Department of Computer Science
Northwestern University
Evanston, Illinois, United States
yingkai.li@u.northwestern.edu

Xiaowei Wu
State Key Lab of IOTSC
University of Macau
Macau, China
xiaoweiwu@um.edu.mo

ABSTRACT
In this paper, we study how to fairly allocate a set of indivisible chores to a number of (asymmetric) agents with additive cost functions. We consider the fairness notion of (weighted) proportionality up to any item (PROPX), and show that a (weighted) PROPX allocation always exists and can be computed efficiently. We also consider the partial information setting, where the algorithms can only use agents’ ordinal preferences. We design algorithms that achieve 2-approximate (weighted) PROPX, and the approximation ratio is optimal. We complement the algorithmic results by investigating the relationship between (weighted) PROPX and other fairness notions such as maximin share and AnyPrice share, and bounding the social welfare loss by enforcing the allocations to be (weighted) PROPX.

CCS CONCEPTS
• Theory of computation → Algorithmic game theory.

KEYWORDS
fair allocation, proportionality, partial information, price of fairness

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM Reference Format:
Bo Li, Yingkai Li, and Xiaowei Wu. 2022. Almost (Weighted) Proportional Allocations for Indivisible Chores. In Proceedings of the ACM Web Conference 2022 (WWW ’22), April 25–29, 2022, Virtual Event, Lyon, France. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3485447.3512057

1 INTRODUCTION
Fairness has drawn an increasing concern in broader areas including but not limited to philosophy, politics, law and recently mathematics and computer science. A fundamental problem here is to fairly allocate a set of resources (goods with non-negative utilities) or tasks (chores with non-positive utilities) to a number of agents. To capture the agents’ preferences for the allocation, people mostly study two solution concepts, envy-freeness [27] and proportionality [36]. Informally, an allocation is regarded as envy-free if nobody wants to exchange her items with any other agent in order to increase her utility. Proportionality is a weaker condition under additive valuations, which only requires that each agent has no smaller utility than her proportional share of all items. The traditional study of fair allocation mostly focused on divisible items (such as land and clean water), where an envy-free or proportional allocation exists [1, 6, 25]. However, the problem becomes trickier when the items are indivisible, due to the fact that an exact envy-free or proportional allocation is not guaranteed; for example, consider the situation of allocating one item to two agents.

Accordingly, for indivisible items, an extensively studied subject is to investigate the extent to which the relaxations of these fairness definitions can be satisfied by either designing (approximation) algorithms or identifying hard instances to show the inherent difficulty of the problems so that no algorithm can achieve better performance than a certain threshold. For envy-freeness, two widely studied relaxations are envy-free up to one item (EF1) [33] and envy-free up to any item (EFX) [21], 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. [33] and Bhaskar et al. [17] proved that an EF1 allocation is 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 relatively unfair even when the instance admits a fairer allocation. EFX is proposed to improve fairness guarantee, where the envy is eliminated after removing any item from the bundle. Though EFX is fairer, it is still unknown whether such an allocation is guaranteed to exist or not, except for some special cases. Regarding proportionality, maximin fairness (MMS) [19] is arguably one of the most extensively studied relaxations. It has been shown that for both goods [32] and chores [8], an exact MMS allocation is not guaranteed to exist, but constant approximation algorithms are known. Proportionality up to one item (PROP1), which is weaker than EF1 and thus inherits many good properties of EF1, has been studied in [7, 24]. Unfortunately, Aziz et al. [7] pointed out that a proportional up to any item (PROPX) allocation is not guaranteed to exist for goods.

We follow this research trend and study PROPX allocations for indivisible chores with additive valuations. PROPX allocation for chores was first proposed in the recent survey by Moulin [34] under the name of fair share up to all items (FSX). Moulin also proposed an algorithm to compute an FSX allocation. In the current
work, we provide a relatively comprehensive study for PROPX, including existence, computation, and price of fairness. Moreover, we are interested in two advanced settings, namely, allocation with asymmetric agents and approximation with ordinal preferences.

First, in many practical scenarios, agents do not have the same share (endowment for the case of goods and obligation for chores) in the system. For example, people at leadership positions may be liable to undertake more responsibility than others in a company. This scenario is called asymmetric or weighted case. Algorithms for computing weighted EF1 or approximately weighted MMS allocations have been proposed for goods in [4, 22, 26]. Recently, AnyPrice Share (APS) fairness is introduced by Babaioff et al. [9], which is more suitable than MMS for asymmetric agents. Before the current work, APS fair allocations for chores have not been studied.

Second, motivated by the practical applications where it is hard for the algorithm to collect complete information regarding agents’ cardinal preferences, using partial information to compute approximately fair allocations has attracted more attention in recent years. Ordinal information setting is a typical case where the algorithm only knows each agent’s ranking on the items without cardinal values. Using ordinal preferences to compute approximately MMS allocations has been studied for goods in [3, 28] and for chores in [5].

In this work, we also design algorithms to compute (approximately) PROPX allocations for asymmetric agents and for the setting with only ordinal information.

1.1 Main Results

We study the problem of allocating \( m \) indivisible chores to \( n \) agents, where each agent \( i \) has obligation \( s_i \geq 0 \) on the share of chores she needs to finish, and \( \sum_{i} s_i = 1 \). Based on whether all agents have share \( 1/n \), we call them symmetric (unweighted) or asymmetric (weighted). Informally, an allocation 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. This is because the existence of MMS/APS allocation 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.

To compute PROPX allocations, we provide two algorithms and each has its own merits. The first algorithm is for symmetric agents and is based on the widely studied envy-cycle elimination technique [12, 33], which is recently adapted to chores and called Top-trading envy-cycle elimination [17]. It is proved in [17] that the algorithm always returns an EF1 allocation. We prove a stronger property — when the agents have the same ordinal preference for the items (which is named IDO instances in [31], short for identical ordering), the returned allocation is EFX (and thus PROPX) if we carefully select the items to be allocated in each step. Then we show a general reduction where if we have an algorithm for IDO instances, we can convert it to handle general instances while guaranteeing (weighted) PROPX. Similar reductions are widely used in the computation of approximate MMS allocations [13, 18, 31]. Finally, we note that Barman and Krishnamurthy [13] proved that the envy-cycle elimination algorithm also ensures \( 4/3 \)-approximation of MMS. We summarize our results in this part as follows.

Result 1.1 (Theorem 3.1). For symmetric agents, there is an allocation that is simultaneously PROPX and \( 4/3 \)-approximate MMS. In addition, when the instance is IDO, the allocation is also EFX.

The second algorithm handles agents with arbitrary shares and also optimizes the efficiency of the allocation. To improve the social welfare, intuitively, each item should be allocated to the agent who has lowest cost on that item. We incorporate this idea to the design of the algorithm, bid-and-take, and show that it actually guarantees the tight approximation ratio to the optimal social cost subject to the (weighted) PROPX constraint, i.e., the price of fairness.

Result 1.2 (Theorems 4.1, 6.1, 7.1, 7.2). The allocation returned by the bid-and-take algorithm is (weighted) PROPX and 2-approximate APS. It also achieves the tight approximation ratio to the optimal social cost subject to the PROPX constraint. The tight bound for the price of fairness regarding PROPX is \( \Theta(n) \) for unweighted case, \( \Theta(m) \) for weighted IDO case and unbounded for weighted case.

Last but not least, following the recent works on the partial information setting [3, 5, 28], we study the problem of designing algorithms that only use agents’ ordinal preferences without exact cardinal values. The intuition behind our algorithms 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 (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 tight. A by-product result in this part is that a weighted EF1 allocation exists for IDO instances.

Result 2 (Lemma 5.1 and Theorem 5.2). With ordinal preferences, our algorithms 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.

1.2 Other Related Works

The definition of EFX allocation for goods was first proposed in [21], after which a lot of effort has been devoted to proving its existence. Currently, we only know that an exact EFX allocation exists for some special cases [2, 15, 23, 35], and the general existence is still unknown. Particularly, Plaut and Roughgarden [35] proved that an EFX allocation exists for IDO instances with goods. Our work complements this result by showing an EFX allocation also exists for IDO instances with chores. While most literatures study the special yet important case where agents have equal entitlement or obligation share to the items, there is also a fast growing recent literature on the more general model in which agents may have arbitrary and possibly unequal shares. For example, Farhadi et al. [26] and Aziz et al. [4] adapted MMS to this setting for goods and chores, respectively, and designed approximation algorithms accordingly. Babaioff et al. [9, 10] provided different generalizations
of MMS to this case. A weighted EF1 allocation is known to exist for goods in [22] but unknown for chores. Recently, AnyPrice Share (APS) fairness is introduced by Babaioff et al. [9], where a 3/5-approximation algorithm is designed for goods. Our work complements this work by showing the existence of 2-approximate APS allocation for chores.

Besides fairness, social welfare, which is somehow a competing criterion to fairness, is another important criterion to evaluate allocations. The loss in social welfare by enforcing allocations to be fair is quantitatively measured by price of fairness. Bounding the price of fairness for goods and chores are widely studied in the literature [11, 14, 16, 20, 29, 30]. In this paper, we study the price of fairness for indivisible chores under (weighted) PROPX requirement, and show that our algorithm achieves the optimal ratio.

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 ∈ N has a cost function ci : 2M → R+ ∪ {0}. The cost functions are assumed to be additive in the current work; that is, for any set S ⊆ M, ci(S) = ∑j∈S ci(j). When there is no confusion, we use ci(j) and ci to denote ci({j}). Without loss of generality, we sometimes assume the cost functions are normalized, i.e., ci(M) = 1. An allocation is represented by a partition of the items X = (X1, . . . , Xn), where each agent i obtains Xi, Xi ∩ Xj = ∅ for all i ̸= j and ∪i∈NXi = M. An allocation is called partial if ∪i∈NXi ⊆ M. Let the social cost of the allocation X be sc(X) = ∑i∈N ci(Xi). We are particularly interested in a special setting, identical ordering (IDO), in which all agents agree on the same ranking of the items, i.e., ci1 ≥ · · · ≥ cin for all i ∈ 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 X is envy-free (EF) if ci(Xi) ≤ ci(Xj) for any i, j ∈ N. The allocation is proportional (PROP) if ci(Xj) ≤ PROP1, for any i ∈ N, where PROP1 = (1/n) · ci(M) is agent i’s proportional share for all items.

For normalized cost functions PROP1 = 1/n, ∀i ∈ N.

Definition 2.2 (EF1 and EFX). An allocation X is envy-free up to one item (EF1) if for any i, j ∈ N, there exists e ∈ Xi such that ci(Xi \ {e}) ≤ ci(Xj). The allocation is envy-free up to any item (EFX) if for any i, j ∈ N and any e ∈ Xi, ci(Xi \ {e}) ≤ ci(Xj).

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 α ≥ 1, an allocation X is a-approximate proportional up to one item (α-PROP1) if for any i ∈ N, there exists e ∈ Xi such that ci(Xi \ {e}) ≤ α · PROP1. The allocation is a-approximate proportional up to any item (α-PROPX) if for any i ∈ N and any e ∈ Xi, ci(Xi \ {e}) ≤ α · PROP1. When α = 1, allocation 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 X and any agent i, ci(Xi \ {e}) ≤ ci(Xj) for all e ∈ Xi and j ∈ N. Summing up the inequalities for all j, we have n · ci(Xi \ {e}) ≤ ci(M). Thus X is PROPX.

Finally, we recall the definition of maximin share fairness.

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

\[ \text{MMS}_i = \min_{\pi \in \Pi(M)} \max \{c_i(X_i)\}. \]

For any α ≥ 1, an allocation X is α-approximate maximin share fair (α-MMS) if ci(Xi) ≤ α · MMSi for all i ∈ N. When α = 1, allocation X is MMS fair.

3 WARM-UP: UNWEIGHTED AGENTS

The existence of PROPX allocations for chores was first proved by Moulin [34] via a novel algorithm. 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 [33] for goods and was adapted to chores in [17] recently. Given any (partial) allocation X = (X1, . . . , Xn), we say that agent i envies j if ci(Xi) > cj(Xj) and most-envies j if ci(Xi) > cj(Xj) and j ∈ arg mink∈N cj(Xk). For any allocation X, we can construct a directed graph GX, called 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 = (i1, . . . , id) is referred to as a top-envy cycle. For any top-envy cycle C, the cycle-swapped allocation X′C is obtained by reallocating bundles backwards along the cycle. That is, X′C = Xi if i is not in C, and

\[ X′_{ij} = \begin{cases} X_{ij} \quad &\text{for all } 1 \leq j \leq d - 1, \\ X_{ij} &\text{for } j = d. \end{cases} \]

The algorithm works by assigning, at each step, an unassigned item with largest 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 GX does not have a sink, then it must have a cycle [17]. 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.

Algorithm 1 is the same with the one designed in [12, 17] except that we allocate the item with largest cost to the sink agent at each step. It is proved in [17] that no matter which item is allocated, the returned allocation is EF1. In the following, we show a stronger
Algorithm 1: Top-trading Envy Cycle Elimination

1. **Input:** IDO instance with \( c_{i1} \geq c_{i2} \geq \cdots \geq c_{im} \) for all \( i \in N \).
2. Initialize: \( X = (X_1, \ldots, X_n) \) where \( X_i \leftarrow \emptyset \) for all \( i \in N \).
3. **for** \( j = 1, 2, \ldots, m \) **do**
   4. **if** there is no sink in \( G_X \) **then**
      5. Let \( C \) be any cycle in \( G_X \).
      6. Reallocate the items according to \( X^C \) (i.e., the cycle-swapped allocation).
      7. Choose a sink \( k \) in the graph \( G_X \) and update \( X_k \leftarrow X_k \cup \{j\} \).
6. **Output:** Allocation \( X = (X_1, \ldots, X_n) \).

We first adapt our solution concepts to the weighted setting. The weighted proportionality of each agent is \( \text{WPROP}_i = s_i \cdot c_i(M) \).

**Algorithm 2:**

1. **Input:** IDO instance with \( c_{i1} \geq c_{i2} \geq \cdots \geq c_{im} \) for all \( i \in N \).
2. Initialize: \( X = (X_1, \ldots, X_n) \) where \( X_i \leftarrow \emptyset \) for all \( i \in N \).
3. **for** \( j = 1, 2, \ldots, m \) **do**
   4. **if** there is no sink in \( G_X \) **then**
      5. Let \( C \) be any cycle in \( G_X \).
      6. Reallocate the items according to \( X^C \) (i.e., the cycle-swapped allocation).
      7. Choose a sink \( k \) in the graph \( G_X \) and update \( X_k \leftarrow X_k \cup \{j\} \).
6. **Output:** Allocation \( X = (X_1, \ldots, X_n) \).

**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 \( s_i \geq 0 \), and \( \sum_{i \in N} s_i = 1 \). Intuitively, \( s_i \) represents how much 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 \( \text{WPROP}_i = s_i \cdot c_i(M) \).

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

**Lemma 4.1.** If there exists a polynomial time algorithm that given any IDO instance computes an \( \alpha \)-WPROP allocation, then there exists a polynomial time algorithm that given any instance computes an \( \alpha \)-WPROP allocation.

Actually, Lemma 4.1 is in the same spirit with the counterpart reductions in [13, 18, 31] which are designed for (unweighted) MMS. Although their results do not directly work for "up to one" relaxations, using the same technique, we show how to extend it to PROPX allocations and weighted settings. It deserves to note that the reduction does not need to access the cardinal costs and thus holds for the ordinal setting as well.

**4.2 Bid-and-Take Algorithm**

Note that the top-trading envy cycle elimination algorithm is not able to compute a WPROPX allocation for 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 inactive her.

**Algorithm 2: Bid-and-Take**

1. **Input:** IDO instance with \( c_{i1} \geq \cdots \geq c_{im} \) and \( c_i(M) = 1 \).
2. Initialize: \( X_i \leftarrow \emptyset \) for all \( i \in N \), and shares of agents \( 0 \leq s_1 \leq s_2 \leq \cdots \leq s_n \) and \( \sum_{i \in N} s_i = 1 \).
3. **for** \( j = 1, 2, \ldots, m \) **do**
   4. Let \( X_i \leftarrow \emptyset \) for all \( i \in N \) and \( A \leftarrow N \) be the set of active agents.
   5. **for** \( j = 1, 2, \ldots, m \) **do**
      6. **if** \( c_i(X_i) > s_i \) **then**
         7. Let \( i \in \arg \min_{e \in A} \{c_i(e)\} \), and set \( X_i \leftarrow X_i \cup \{j\} \).
         8. Let \( A \leftarrow A \setminus \{j\} \).
6. **Output:** Allocation \( X = (X_1, \ldots, X_n) \).

**Lemma 4.2.** Algorithm 2 returns a WPROPX allocation in polynomial time for IDO instances.

**Proof.** Note that in Algorithm 2, an agent is turned into 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 $X = (X_1, \ldots, X_n)$ 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 $A \neq \emptyset$ when considering each item $j \in M$. \hfill \Box

\textbf{Claim 4.1.} At any point, for any active agents $i, i' \in A$, $c_i(X_{i'}) \geq c_{i'}(X_{i'})$.

\textbf{Proof.} Since we allocate each item to the active agent that has smallest cost on the item, we have $c_{ij} \geq c_{i'j}$ for each item $j \in X_i$ (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.

Suppose when we consider some item $j$, 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

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

where the first inequality follows as item $j$ has not been allocated yet, which is a contradiction with $c_i(M) = 1$. \hfill \Box

Combining Lemmas 4.2 and 4.1, we have the following theorem.

\textbf{Theorem 4.1.} There is an algorithm that computes a WPROPX allocation for any weighted instance in polynomial time.

\section{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.

\subsection*{5.1 Unweighted Setting}

To highlight the intuition, we first consider the unweighted case, and present Algorithm 3 that 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 each agent in $N_2$ takes turns to select an item with largest cost from the remaining items.

\textbf{Theorem 5.1.} Algorithm 3 returns a 2-PROPX allocation in polynomial time.

\textbf{Proof.} It is straightforward that the algorithm runs in polynomial time. 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 $X_1, \ldots, X_{|N_1|}$ 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

$$c_i(e_1) \leq c_i(X_i) \quad \text{for all } i \in N_1. \quad (1)$$

Moreover, as the items $\{\lfloor n/2 \rfloor + 1, \ldots, m\}$ 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

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

where the last inequality follows from Inequality (1). Thus

$$c_i(X_i) = c_i(e_1) + c_i(X_i \setminus \{e_1\})$$

$$\leq c_i(e_1) + \frac{1}{|N_2|} \left(1 - (|N_1| + 1) \cdot c_i(e_1)\right)$$

$$= \frac{1}{|N_2|} + \left(1 - \frac{|N_1| + 1}{|N_2|}\right) \cdot c_i(e_1)$$

$$\leq \frac{1}{|N_2|} \leq \frac{2}{|N_2|} = 2 \cdot PROPX,$$

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

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

\textbf{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 $c_{1i} \geq \cdots \geq c_{mi}$ for both $i \in \{1, 2\}$. Without loss of generality, suppose item 1 is allocated to agent 1. If $|X_1| > 1$, consider the cardinal costs for agent 1, $c_{11} = 1$ and $c_{1j} = 0$ for all $j > 1$, and thus $c_1(X_1 \setminus \{e\}) = 2 \cdot \text{PROP}_1$ for any $e \in X_1 \setminus \{1\}$. If $|X_1| = 1$, consider the cardinal costs for agent 2, $c_{2j} = 1/m$ for all $j$, and thus $c_2(X_2 \setminus \{e\}) = 1 - 2/m \approx 2 \cdot \text{PROP}_2$ 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

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

We partition the agents into two groups: $N_1 = \{i^*\}$ and $N_2 = N \setminus N_1$. Let $w_i = \sum_{e \in N_i} s_i$. 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 $s_i \geq 1/2(i^* + 1)$ for all $i \in N_2$ because otherwise $s_{i+1} < 1/2(i^* + 1)$, 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 $j \in M$. Recall that these are the $i^*$ items with the maximum costs. Let $M_L = \{1, 2, \ldots, i^*\}$ be these items, and $M_R = M \setminus M_L$ 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 E1 for indivisible goods in [22]. In Lemma 5.2, we prove that the weighted round robin algorithm also ensures weighted E1 for IDO instance with chores.

Algorithm 4: Ordinal Approximate WPROPX Allocation

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

Our main result relies on the following technical lemma.

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

$$\frac{c_i(X_i \setminus \{i\})}{s_i} \leq \frac{c_i(X_i)}{s_i}.$$

Proof. Suppose $X_i = \{e_1, e_2, \ldots, e_k\}$, where $c_{ie_1} \geq c_{ie_2} \geq \cdots \geq c_{ie_k}$. Recall that $e_1 = i$. We define a real-valued function $\rho : (0, k/s_i) \rightarrow \mathbb{R}^+$ as follows:

$$\rho(a) = c_{ie_r}, \text{ for } \frac{t-1}{s_i} < a \leq \frac{t}{s_i} \text{ and } t \in [k].$$

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

$$\frac{c_{ie_r}}{s_i} = \int_{\frac{t-1}{s_i}}^{\frac{t}{s_i}} \rho(a) da \quad \text{and} \quad \frac{c_i(X_i \setminus \{i\})}{s_i} = \int_{\frac{1}{s_i}}^{\frac{k}{s_i}} \rho(a) da.$$

Similarly, assume $X_j = \{e_1', \ldots, e_k'\}$, where $c_{ie_r'} \geq \cdots \geq c_{ie_k'}$, and define

$$\rho'(a) = c_{ie_r'}, \text{ for } \frac{t-1}{s_j} < a \leq \frac{t}{s_j}.$$

By definition we have

$$\frac{c_i(X_j)}{s_j} = \int_0^{\frac{k'}{s_j}} \rho'(a) da.$$

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

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

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

Next we show that $\rho(a) \leq \rho'(a - 1/s_i)$. Consider the round when $|X_i|/s_i$ reaches $a$. Suppose item $e_r$ is allocated to agent $i$ at this round, i.e., $\rho(a) = c_{ie_r}$. Note that in this round, we must have $|X_i|/s_i \geq a - 1/s_i$. Because otherwise when item $e_r$ is considered we have $|X_i|/s_i < a - 1/s_i \leq |X_j|/s_j$, which means that item $e_r$ should not be allocated to agent $i$. In other words, the event “$|X_i|/s_i$ reaches $a - 1/s_i$” happens before the event “$|X_j|/s_i$ reaches $a$”. Since items are allocated from the most costly to the least costly, we have $\rho(a) = c_{ie_r} \leq \rho'(a - 1/s_i)$.

Combining the above discussion, we have

$$\frac{c_i(X_i \setminus \{i\})}{s_i} = \int_0^{\frac{k}{s_i}} \rho(a) da \leq \int_0^{\frac{k-1}{s_i}} \rho'(a - \frac{1}{s_i}) da$$

$$\leq \int_0^{\frac{k-1}{s_i}} \rho'(a) da \leq \int_0^{\frac{k'}{s_j}} \rho'(a) da = \frac{c_i(X_j)}{s_j},$$

which proves the lemma. □

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

Theorem 5.2. Algorithm 4 computes a 2-WPROXP allocation in polynomial time for any given weighted instance.

Proof. As before, it suffices to show that the allocation is 2-WPROXP for $N_2$, i.e., $c_i(X_i) \leq 2 \cdot s_i$ 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

$$\sum_{j \in N_2} \frac{s_j}{s_i} \cdot c_i(X_i \setminus \{i\}) \leq c_i(X_i \setminus \{i\}) + \sum_{j \in N_2 \setminus \{i\}} c_i(X_j) = c_i(M_S) - c_{ii}.$$
We start with symmetric agents and MMS fairness. Before presenting our results, we recall the following inequality.

\[
\text{MMS}_i \geq \max_{j \in M} \{c_{ij}\}, \forall i \in N.
\]

**Lemma 6.1.** Any PROXP allocation is 2-MMS.

**Proof.** We will prove a stronger argument here: for any PROXP allocation \(X\) and for any agent \(i\), either \(|X_i| \leq 1\) or \(c(X_i) \leq 2\cdot \text{PROP}_i\), for any \(e \in X_i\). Then by Equation (2), Lemma 6.1 holds. For any agent \(i\), if \(|X_i| \leq 1\), the claim holds trivially. If \(|X_i| \geq 2\), letting \(e_i = \min_{e \in X_i \setminus \{e_i\}} c_i(e)\), we have

\[
c_i(X_i \setminus \{e_i\}) \leq \text{PROP}_i,
\]

and

\[
c_i(e_i) \leq c_i(X_i \setminus \{e_i\}) \leq \text{PROP}_i.
\]

Thus \(c_i(X_i) = c_i(X_i \setminus \{e_i\}) + c_i(e_i) \leq 2 \cdot \text{PROP}_i\). \(\Box\)

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 PROXP but only 2-MMS.

**Lemma 6.2.** There exists an MMS allocation that is \(\Theta(n)\)-PROXP.

**Proof.** Consider an instance with \(n\) agents and \(m = n\) items where \(n\) is sufficiently large. In this instance all agents have identical cost function for the items. For each agent \(i\), let \(c_{ij} = n - 1\) and \(c_{ij} = 1\) for items \(j = 2, \ldots, n\). Thus

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

Consider an allocation where \(X_i = \{2, \ldots, n\}\) is allocated to some agent \(i\). Note that \(c_i(X_i \setminus \{j\}) \leq s_i\) for any \(j \in X_i\). Thus \(c_i(X_i) \leq s_i + c_{ij} \leq s_i + \max_{e \in M} c_{ie} \leq 2 \cdot \text{APS}_i\), where the last inequality follows from Lemma 6.4.

To see the second claim, we recall the example in Lemma 6.2. In that example there are \(n\) agents and \(m = n\) items, and for all agent \(i \in N\) we have \(c_{i1} = n - 1\) and \(c_{ij} = 1\) for items \(j = 2, \ldots, n\). Thus

\[
\text{APS}_i \geq \max_{j \in M} \{c_{ij}\} = n - 1 \quad \text{and} \quad \text{PROP}_i = \frac{2(n - 1)}{n}.
\]

Thus allocating \(\{2, \ldots, n\}\) to some agent \(i\) is APS to her but has \(\Theta(n)\) approximation regarding PROXP.

**Theorem 4.1** and Lemma 6.3 immediately imply the following.

**Theorem 6.1.** There is an algorithm that computes a 2-APS fair allocation for any weighted instance in polynomial time.
### 7 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) PROXP 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 \( \Omega(I) \) be the set of all WPROXP allocations for instance \( I \). The price of fairness is defined as the worst-case ratio between the optimal (minimum) social cost \( \text{opt}(I) \) without any constraints and the social cost under WPROXP allocations:

\[
PoF = \max_{I} \min_{X \in \Omega(I)} \frac{\text{sc}(X)}{\text{opt}(I)}.
\]

Note that for any instance \( I \), \( \text{opt}(I) \) 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 agents \( i \in N \) 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 WPROXP allocations cannot.

**Lemma 7.1.** Letting \( X \) be the allocation returned by Algorithm 2, we have \( \text{sc}(X) \leq 1 \).

**Proof.** Let \( i \in N \) be the agent that receives the last item. 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 \( \text{sc}(X) = \sum_{j \in N} c_i(X_j) \leq \sum_{j \in N} c_i(X_j) = c_i(M) = 1 \). \( \square \)

We show that Algorithm 2 achieves the optimal PoF.

**Theorem 7.1.** For the unweighted case, the PROXP 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 \( I \) for which any PROXP allocation \( X \) satisfies \( \text{sc}(X) \geq (n/6) \cdot \text{opt}(I) \). In \( I \), we have \( n \) agents and \( m = n \) items with cost functions shown in the table below.

| Agent | 1 | 2 | \( n-1 \) | \( n \) |
|-------|---|---|---------|------|
| \( c_{i1} \) | \( 2/n^2 \) | \( 2/n^2 \) | \( 1 - 2(n-1)/n^2 \) | \( 1/n \) |
| \( c_{i2} \) | \( 1/n \) | \( 1/n \) | \( 1/n \) | \( 1/n \) |
| \( \vdots \) | \( \vdots \) | \( \vdots \) | \( \vdots \) | \( \vdots \) |
| \( c_{in} \) | \( 1/n \) | \( 1/n \) | \( 1/n \) | \( 1/n \) |

**Proof.** For the above instance we have

\[
\text{opt}(I) = (n - 1) \cdot \frac{3}{n^2} + \frac{3}{n} < \frac{3}{n}.
\]

However, any PROXP allocation \( X \) 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

\[
\text{sc}(X) \geq \left( \frac{n}{2} + 1 \right) \cdot \frac{2}{n^2} + \left( \frac{n}{2} - 1 \right) \cdot \frac{1}{n} > \frac{1}{2} \geq \frac{n}{6} \cdot \text{opt}(I).
\]

Next we show that for any unweighted instance \( I \), the allocation \( X \) computed by Algorithm 2 satisfies \( \text{sc}(X) \leq n \cdot \text{opt}(I) \). By Lemma 7.1, it suffices to consider the case when \( \text{opt}(I) < 1/n \). We show that in this case, we have \( \text{sc}(X) = \text{opt}(I) \). 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 social optimal allocation. Since \( \text{opt}(I) < 1/n \), Algorithm 2 never turns any agent into inactive, which implies that \( \text{sc}(X) = \text{opt}(I) \). \( \square \)

We note that the hard instance to show Theorem 7.1 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 7.2.** For the weighted case, we have unbounded PoF. For IDO instances, Algorithm 2 computes a WPROXP 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 \( I \) (with \( s_1 = 1 - e^2 \) and \( s_2 = e^2 \)) shown in the table below. It is easy to see that \( \text{opt}(I) = 2e \). However, since any WPROXP allocation \( X \) allocates at most one item to agent 2, we have \( \text{sc}(X) \geq 1/2 \). Since \( e > 0 \) can be arbitrarily close to 0, we have an unbounded PoF for the weighted non-IDO instances.

| Agent | \( c_{i1} \) | \( c_{i2} \) | \( c_{i3} \) | \( \cdots \) | \( c_{im} \) |
|-------|---|---|---|---|---|
| 1 | 0.5 | 0.5 | 0 | \( \cdots \) | 0 |
| 2 | \( 1/m \) | \( 1/m \) | \( 1/m \) | \( \cdots \) | \( 1/m \) |

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

**Proof.** It is easy to see that \( \text{opt}(I) = 2/m \). However, for \( s_1 = 1 - 1/m^2 \) and \( s_2 = 1/m^2 \), since any WPROXP allocation \( X \) allocates at most one item to agent 2, we have \( \text{sc}(X) \geq 1/2 \geq m/4 \cdot \text{opt}(I) \). Finally, we show that the weighted IDO instances the allocation \( X \) returned by Algorithm 2 satisfies \( \text{sc}(X) \leq m \cdot \text{opt}(I) \). By Lemma 7.1, \( \text{sc}(X) \leq 1 \). Moreover, for IDO instances,

\[
\text{opt}(I) = \sum_{j \in M} \min_{i \in N} c_{ij} \geq \min_{i \in N} c_{i1} \geq \frac{1}{m} \geq \frac{1}{m} \cdot \text{sc}(X).
\]

Hence allocation \( X \) achieves the asymptotically optimal PoF. \( \square \)

### 8 CONCLUSION

In this paper, we studied the fair allocation of indivisible chores under the fairness notion of PROXP. We showed that PROXP 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) PROXP 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 effort. To name a few, as we have discussed, the existence or approximation of EFX and weighted EF1 allocations are less explored for chores than those for goods. Furthermore, we proved that any WPROXP allocation is 2-approximate APS, but it does not have good guarantee for weighted MMS defined in [4]. It is still unknown whether weighted MMS admits constant approximations. Finally, we believe it is an important problem to investigate the compatibility between PROXP and efficiency notions such as Pareto optimality.
REFERENCES

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

[2] Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, Alexandros Hol- lender, and Alexandros A. Voudouris. 2020. Maximum Nash Welfare and Other Stories About EFk. In *IJCAI*. ijcai.org, 24–30.

[3] Georgios Amanatidis, Georgios Birmpas, and Evangelos Markakis. 2016. On Truthful Mechanisms for Maximin Share Allocations. In *IJCAI*. ijcai.org, 31–37.

[4] Haris Aziz, Hau Chan, and Bo Li. 2019. Weighted Maximax Fair Share Allocation of Indivisible Chores. In *IJCAI*. ijcai.org, 46–52.

[5] Haris Aziz, Bo Li, and Xiaowei Wu. 2020. Approximate and Strategyproof Maximin Share Allocation of Chores with Ordinal Preferences. *CoRR* abs/2012.13884 (2020).

[6] Haris Aziz and Simon Mackenzie. 2016. A Discrete and Bounded Envy-Free Cake Cutting Protocol for Any Number of Agents. In *FOCS*. IEEE Computer Society, 416–427.

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

[8] Haris Aziz, Gerhard Rauchecker, Guido Schreyer, and Toby Walsh. 2017. Algorithms for Max-Min Share Fair Allocation of Indivisible Chores. In *AAAI*. AAAI Press, 335–341.

[9] Moshe Babaioff, Tamer Ezra, and Uriel Feige. 2021. Fair-Share Allocations for Agents with Arbitrary Entitlements. *EC* (2021).

[10] Moshe Babaioff, Noam Nisan, and Inbal Talgam-Cohen. 2016. A polynomial-time algorithm for computing a Pareto optimal and almost proportional allocation. *Oper. Res. Lett.* 48, 5 (2020), 573–578.

[11] Siddharth Barman, Umang Bhaskar, and Nisarg Shah. 2020. Optimal Bounds on the Price of Fairness for Indivisible Goods. In *WINE (Lecture Notes in Computer Science, Vol. 12495)*. Springer, 356–369.

[12] Siddharth Barman, Arpita Biswas, Sanath Kumar Krishna Murthy, and Yadati Narahari. 2018. Groupwise Maximax Fair Allocation of Indivisible Goods. In *AAAI*. AAAI Press, 917–924.

[13] Siddharth Barman, Arpita Biswas, Sanath Kumar Krishnamurthy. 2017. Approximation Algorithms for Maximin Fair Division. In *EC*. ACM, 647–664.

[14] Xiaohui Bei, Xinhang Lu, Pasin Manurangsi, and Warut Suksompong. 2019. The Price of Fairness for Indivisible Goods. In *IJCAI*. ijcai.org, 81–87.

[15] Ben Berger, Avi Cohen, Michal Feldman, and Amos Fiat. 2021. (Almost Full) EFX Exists for Four Agents (and Beyond). *CoRR* abs/2102.10654 (2021).

[16] Dimitris Bertsimas, Vivek F. Farias, and Nikolaos Trichakis. 2011. The Price of Fairness. *Oper. Res.* 59, 1 (2011), 17–31.

[17] Umang Bhaskar, A. R. Sricharan, and Rohit Vaish. 2021. On Approximate Envy-Freeness for Indivisible Chores and Mixed Resources. In *APPROX-RANDOM (LIPIcs, Vol. 207)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 1:1–1:23.

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

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

[20] Ioannis Caragiannis, Christos Karamanis, Panagiotis Kanellopoulos, and Maria Kyropoulou. 2009. The Efficiency of Fair Division. In *WINE (Lecture Notes in Computer Science, Vol. 5929)*. Springer, 475–482.

[21] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, and Junxing Wang. 2019. The Unreasonable Fairness of Maximum Nash Welfare. *ACM Trans. Economics and Comput.* 7, 3 (2019), 12:1–12:32.

[22] Mithun Chaikarborty, Aayumi Igarashi, Warut Suksompong, and Yair Zick. 2020. Weighted Envy-Freeness in Indivisible Item Allocation. In *AAMAS International Foundation for Autonomous Agents and Multiagent Systems*, 231–239.

[23] Bhaskar Ray Chaudhury, Jugal Garg, and Kurt Mehlhorn. 2020. EFX Exists for Three Agents. In *EC*. ACM, 1–19.

[24] Vincent Conitzer, Rupert Freeman, and Nisarg Shah. 2017. Fair Public Decision Making. In *EC*. ACM, 629–646.

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

[26] Alirea Farhadi, Mohammad Ghodsi, Mohammad Taghi Hajiaghayi, S´ebastien Lahtee, David M. Pennock, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. 2019. Fair Allocation of Indivisible Goods to Asymmetric Agents. *J. Artif. Intell. Res.* 64 (2019), 1–20.

[27] Duncan Foley. 1967. Resource Allocation and the Public Sector. *Yale Economic Essays* (1967), 45–98.

[28] Daniel Halpern and Nisarg Shah. 2021. Fair and Efficient Resource Allocation with Partial Information. *CoRR* abs/2105.10064 (2021).

[29] Sandy Heydrich and Rob van Stee. 2015. Dividing connected chores fairly. *Theor. Comput. Sci.* 593 (2015), 51–62.

[30] Felix Höhne and Rob van Stee. 2021. Allocating contiguous blocks of indivisible chores fairly. *Information and Computation* (2021), 104739.

[31] Xin Huang and Pinyan Lu. 2019. An algorithmic framework for approximating maximin share allocation of chores. *CoRR* abs/1907.04565 (2019).

[32] David Kurokawa, Ariel D. Procaccia, and Junxing Wang. 2018. Fair Enough: Guaranteeing Approximate Maximin Shares. *J. ACM* 65, 2 (2018), 8:1–8:27.

[33] Richard J. Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. 2004. The Price of Fairness for Indivisible Goods. In *WINE (Lecture Notes in Computer Science, Vol. 3174)*. Springer, 356–369.

[34] Hervé Moulin. 2019. Fair Division in the Internet Age. *Annual Review of Economics* 11 (2019), 1–37.

[35] Benjamin Plaut and Tim Roughgarden. 2020. Almost Envy-Freeness with General Valuations. *SIAM J. Discret. Math.* 34, 2 (2020), 1039–1068.

[36] Hugo Steinhaus. 1948. *The problem of fair division*. *Econometrica* 16 (1948), 101–104.
APPENDIX

A PROOF OF LEMMA 3.1

Proof. Envy cycle elimination algorithm runs in polynomial time, which has been proved in [17, 33]. 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 X be a partial and EFX allocation at the beginning of any round in Algorithm 1. Let X0 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 A.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) ≤ c_i(X_j) for all j ∈ N. Let e be the item with largest cost in X0, 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') ≥ c_i(e) for all e' ∈ X_i. Thus for any e' ∈ X_i ∪ {e} and any j ≠ i,

\[ c_i(X_i ∪ \{e\} \setminus \{e'\}) ≤ c_i(X_i ∪ \{e\} \setminus \{e'\}) = c_i(X_i) ≤ c_i(X_j). \]

Thus Claim A.1 holds.

Claim A.2. Resolving a top-trading envy cycle preserves EFX.

Suppose we reallocate the bundles according to a top-envy cycle C = (i_1, ..., i_k) 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 is 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 X, and hence the cycle-swapped allocation is EF for agent i. Thus Claim A.2 holds.

Combining the two claims, at the end of each round, the partial allocation remains EFX.

B PROOF OF LEMMA 4.1

Proof. In the following, we explicitly write \( I = (N, s, M, e) \) to denote an instance with item set \( M \), agent set \( N \), weight vector \( s = (s_1, ..., s_n) \), and cost functions \( e = (c_1, ..., c_n) \). Given any instance \( I = (N, s, M, e) \), we construct an IDO instance \( I' = (N, s, M, e') \) where \( e' = (c'_1, ..., c'_n) \) is defined as follows. Let \( \sigma_i(j) \in M \) be the j-th most costly item under cost function \( c_i \). Let

\[ c'_{ij} = c_i(\sigma_i(j)). \]

Thus with cost functions \( e' \), the instance \( I' \) is IDO, in which all agents i has

\[ c'_{ij} = c_{ij}' \geq \cdots \geq c_{im}'. \]

Then we run the algorithm for IDO instances on instance \( I' \), and get an \( \alpha \)-WPROPX allocation \( X' \) for \( I' \). By definition, for all agent \( i \in N \) we have

\[ c'_i(X'_i - e) \leq \alpha \cdot s_i, \quad \forall e \in X'_i. \]

In the following, we use \( X' \) to guide us on computing a \( \alpha \)-WPROPX allocation \( X \) for instance \( I \).

Recall that in the IDO instance \( I' \), for all agents, item 1 has the maximum cost and item 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 item \( j \) from \( m \) to 1, we let the agent \( i \) that receives item \( j \) under allocation \( X' \), i.e., \( j \in X'_i \), pick her favourite unallocated item. Note that the order of items are well-defined in the IDO instance \( I' \). 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 \( f_i : X_i \rightarrow X'_i \) such that for any item \( e \in X_i \), we have \( c_{ie} \leq c'_{ij} \), and hence the cycle-swapped allocation is EF for agent i. Thus Claim A.2 holds.

Combining the two claims, at the end of each round, the partial allocation remains EFX.