How to Fairly Allocate Easy and Difficult Chores

Soroush Ebadian
University of Toronto
Toronto, Ontario, Canada
soroush@cs.toronto.edu

Dominik Peters
University of Toronto
Toronto, Ontario, Canada
dominik@cs.toronto.edu

Nisarg Shah
University of Toronto
Toronto, Ontario, Canada
nisarg@cs.toronto.edu

ABSTRACT

A major open question in fair allocation of indivisible items is whether there always exists an allocation of chores that is Pareto optimal (PO) and envy-free up to one item (EF1). We answer this question affirmatively for the natural class of bivalued utilities, where each agent partitions the chores into easy and difficult ones, and has cost $p > 1$ for chores that are difficult for her and cost 1 for chores that are easy for her. Such an allocation can be found in polynomial time using an algorithm based on the Fisher market.

We also show that for a slightly broader class of utilities, where each agent i can have a potentially different integer p_i, an allocation that is maximin share fair (MMS) always exists and can be computed in polynomial time, provided that each p_i is an integer. Our MMS arguments also hold when allocating goods instead of chores, and extend to another natural class of utilities, namely weakly lexicographic utilities.

KEYWORDS

Fair Division; Envy-Freeness; Pareto Optimality; Maximin Share

ACM Reference Format:
Soroush Ebadian, Dominik Peters, and Nisarg Shah. 2022. How to Fairly Allocate Easy and Difficult Chores. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

Fair allocation of collective resources and burdens between agents is a fundamental task in multi-agent systems. Everyday applications include splitting an estate between heirs or joint assets between a divorcing couple (resources), or splitting work shifts between staff or household chores between roommates (burdens).

We are interested in indivisible resources and burdens (i.e., ones that cannot be subdivided). Let $\mathcal{M}$ be the set of such *items*. Following a canonical model, we assume that each agent i has a valuation $v_i(r)$ for each item $r \in \mathcal{M}$. This gives rise to an *additive* utility function over bundles of items: Agent i's utility for a bundle $S \subseteq \mathcal{M}$ is $v_i(S) = \sum_{r \in S} v_i(r)$. Items are called *goods* if all agents have non-negative valuations for them, and *chores* if all agents have non-positive valuations for them. We will only study cases where either all items are goods, or all items are chores. The goal is to find an *allocation* $\mathbf{x}$, which is a partition of the set $\mathcal{M}$ of items between the agents, with $\mathbf{x}_i$ denoting the bundle allocated to agent i. An allocation is *efficient* or *Pareto optimal* (PO) if there is no other allocation $\mathbf{y}$ which every agent i weakly prefers to $\mathbf{x}$ (i.e., $v_i(\mathbf{y}_i) \geqslant v_i(\mathbf{x}_i)$), and for which at least one of these inequalities is strict. We are interested in finding allocations that are efficient and also *fair*. In particular, we will look at restricted classes of utilities that allow us to guarantee stronger fairness axioms than the state of the art for general additive utilities.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

1.1 Envy-Freeness Up To One Item (EF1)

Perhaps the most compelling fairness guarantee from the literature is *envy-freeness* (EF) [19, 20], which demands that no agent envy another agent (i.e., $v_i(\mathbf{x}_i) \geqslant v_i(\mathbf{x}_j)$ for all agents i, j). However, it is easy to see that envy-freeness cannot be guaranteed; if we are allocating a single item between two agents, then one will necessarily envy the other. In response, the literature has turned to relaxations which require that agents not envy others by too much. A particularly appealing axiom is called *envy-freeness up to one item* (EF1) [15], which demands that envy between any two agents be avoidable by the removal of a single item from the bundle of one of the two agents. For allocating goods, Caragiannis et al. [16] show that an elegant rule called *maximum Nash welfare* (MNW) satisfies EF1 and PO simultaneously. Informally, this rule maximizes the product of utilities of the agents for their assigned bundles, i.e., $\prod_i v_i(\mathbf{x}_i)$. Due to its attractive properties, MNW has been deployed to the popular fair division website Spliddit.org, where it has been used by more than 10,000 people for applications such as dividing estates and settling divorces [35]. Unfortunately, MNW has no natural equivalent for chores, and whether an EF1 and PO allocation of chores always exists has remained a major open question.

To make progress in resolving this problem, we look towards restricted families of utility functions. An example is the class of *binary* utilities, in which all valuations are in $\{0, -1\}$. For allocating goods, the corresponding class of $\{0, 1\}$-utilities is interesting and well-understood [10, 28]. But for allocating chores, this class is trivial: first allocate any chore for which some agent has utility 0 to such an agent; then all agents have utility -1 for all remaining chores, and we can allocate them as equally as possible to obtain an EF1 + PO allocation.

A larger class is that of *bivalued* utilities, where all valuations are in $\{a, b\}$, for some fixed $0 > a > b$. The corresponding class for goods (with $0 < a < b$) has already received significant attention in the literature [1, 5, 23], where it has been used to achieve fairness guarantees stronger than EF1 [2]. This class seems interesting for practical applications: when eliciting agent preferences, it is often cumbersome for agents to submit exact numerical utilities. Instead, it is much easier to ask each agent to classify chores into easy and difficult ones, with an interface familiar from approval voting. Then, one can fix reasonable values of a and b, and assume that all agents have utility a for the chores they consider to be easy and b for the ones they consider to be difficult.

For our results, scaling an agent's utilities multiplicatively makes no difference. Hence, bivalued utilities for chores can also be thought of as having utilities in $\{-1, -p\}$ for some number $p = \frac{b}{a} > 1$ (or $\{1, p\}$ for goods). Our main contribution is to show that EF1 and PO allocations of chores always exist under bivalued utilities, and that such an allocation can be found in polynomial time. We obtain this result via an algorithm based on Fisher markets. We borrow ideas from the existing Fisher-market-based algorithm for finding an EF1 and PO allocation of goods [9, 23], but combine it with a more intricate analysis and new techniques that are key to making the algorithm work for chores. In simultaneous independent work, Garg et al. [25] obtained the same result, also via Fisher markets.

1.2 Maximin Share Fairness (MMS)

In addition to envy-freeness up to one good (EF1), we consider another popular relaxation of envy-freeness called *maximin share fairness* (MMS) [15]. This notion wants to give each agent i at least as much utility as the maximum that i can achieve by partitioning the items into n bundles and receiving her least preferred bundle from that partition. For general additive valuations, an MMS allocation may not exist for goods [32] or for chores [7]. Thus, we again turn to restricted utility classes that allow us to guarantee MMS.

We first consider *personalized bivalued utilities*, where the valuations of each agent i for the chores (resp., goods) lie in $\{-1, -p_i\}$ (resp., $\{1, p_i\}$) for some $p_i > 1$. The value p_i can differ between agents. To elicit such valuations, one can ask each agent i to first partition the chores into easy and difficult ones (resp., goods into ordinary and preferred ones) using an approval interface, and then submit a number p_i indicating how many easy chores they would do instead of a single difficult one (resp., how many ordinary goods they would take in place of a single preferred one). We show that for personalized bivalued utilities, for both goods and chores, an allocation satisfying MMS always exists and can be found in polynomial time, provided that p_i is an *integer* for each agent i. Integrality would be the natural outcome of the aforementioned elicitation. Whether MMS can be guaranteed for non-integral p_i remains an open question. For (non-personalized) bivalued utilities, we can compute in polynomial time an MMS allocation that is also PO.

We also prove the existence of MMS allocations for *weakly lexicographic* utilities, for both goods and chores. Weakly lexicographic utilities are a natural assumption if valuations are elicited by a system that asks agents to rank the items in order of desirability, allowing for ties. The defining assumption is that an agent likes each good (resp., dislikes each chore) more than all strictly less preferred goods (resp., more preferred chores) combined. Such utility functions, which we refer to as weakly lexicographic utilities, have been considered in the literature [4]. We prove that for these utilities, an allocation that satisfies both MMS and PO always exists and can be computed in polynomial time. Hosseini et al. [29] prove this for the special case of allocating goods under (strictly) *lexicographic* utilities (in which there are no ties); our result extends theirs to allocating goods or chores under weakly lexicographic utilities.

Both of our MMS existence results depend on a simple algorithm for computing MMS values (i.e., the utility value guaranteed by the MMS property on a given instance). Computing these values is NP-hard for both goods and chores under general additive valuations (being a special case of the 3-Partition problem [21, p. 224]), but we show that it can be done in polynomial time for *factored* utility functions, which includes both personalized bivalued and weakly lexicographic utilities as special cases. A utility function is factored if the non-zero utility values, say $p_1, \ldots, p_k$, that it uses are such that p_{j+1} is an integer multiple of p_j for each $j \in [k-1]$; examples are $\{1, 2, 6, 12\}$-valuations and $\{0, -1, -5, -45\}$-valuations.

Figure 1 shows the utility classes that we study, together with relevant results, both known and new. Due to space constraints, proofs are omitted and can be found in the full version [18].

1.3 Related Work

Let us summarize a few related threads of work on fair allocation of goods and chores to better contextualize our contributions.

Fisher market. As mentioned earlier, we achieve our main result — an EF1 + PO allocation of chores with bivalued utilities — using the framework of Fisher markets and competitive equilibria. Fisher markets are typically studied for items that are *divisible*, i.e., that can be portioned out fractionally between the agents. In this case, a Fisher market equilibrium allocation exists and is EF + PO [36]. For goods, these allocation happen to be those that maximize the Nash welfare, and they can be computed in strongly polynomial time [17, 34]. For chores, the set of equilibria has a more intricate structure [11] and their computation is an open question [14]; Boodaghians et al. [12] design an FPTAS for this problem. One issue with chore allocation is that neither minimizing nor maximizing the product of agents' costs for their assigned bundles (the equivalent of the Nash welfare objective for goods) yields a desirable allocation. However, Bogomolnaia et al. [11] show that maximizing this objective *subject to PO* yields one of the aforementioned equilibria. Unfortunately, the natural analog of this rule for indivisible chores fails EF1, even for bivalued utilities. Barman et al. [9] adapt Fisher markets to indivisible goods. They use this framework to show that an EF1 + PO allocation can be found in pseudo-polynomial time. Garg and Murhekar [23] improve the running time to strongly polynomial when each agent has at most polynomially many utility levels across all bundles of goods. The Fisher market approach has also been used to obtain efficient allocations that are proportional up to one item (PROP1) for both goods [8] and chores [14].

Factored bivalued utilities and max Nash welfare. The special case of bivalued utilities in which the utility values lie in $\{a, b\}$ for $|a| < |b|$ and b/a is an *integer* (which we refer to as factored bivalued utilities) has been studied in the context of allocating goods. The maximum Nash welfare (MNW) rule is NP-hard to compute for general additive utilities [16], while Barman et al. [10] show that it can be computed in polynomial time for binary ($\{0, 1\}$) utilities. For bivalued utilities, its computability was an open question until recently when Akrami et al. [1] established a surprising dichotomy: it is polynomial-time computable when b/a is an integer (factored bivalued utilities) but NP-hard to compute if a and b are coprime.

MMS. For allocating goods, Kurokawa et al. [32] show that there exists an instance with additive utilities in which no allocation satisfies MMS. This motivates two threads of work. One, similarly to our work, focuses on establishing the existence (and sometimes efficient computability) of MMS allocations under restricted utility classes such as utility functions with identical multisets [13], (strictly) lexicographic utilities [29], and ternary ($\{0, 1, 2\}$) utilities [2]. We

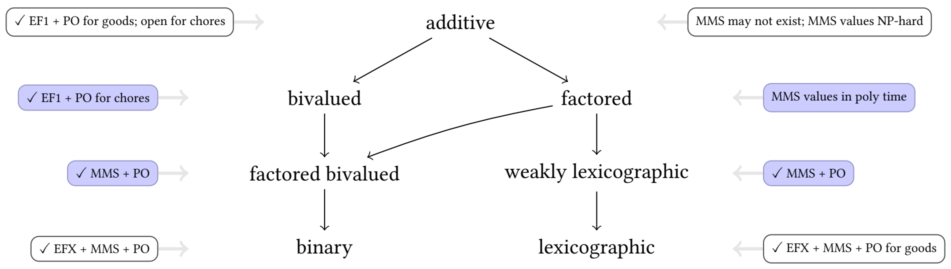

Figure 1: Hasse diagram of valuation classes and results. Shaded blue nodes are new results of this paper, boxed results are known. Checkmarks (✓) denote existence results, which all come with polynomial-time algorithms. Results hold for both goods and chores unless otherwise indicated.

argued in the introduction that 0-utilities can be easily dealt with for chores, so our bivalued result also works for $\{0, -1, -2\}$-utilities, mirroring the result of Amanatidis et al. [2]. The other thread focuses on approximating the MMS guarantee for general additive utilities: the best known multiplicative approximations are (slightly better than) 3/4 for goods [26] and 9/11 for chores [30].

2 PRELIMINARIES

For $k \in \mathbb{N}$, define $[k] = \{1, \ldots, k\}$.

Instances: A *fair division instance* is given by $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$, where $\mathcal{N} = [n]$ is a set of n agents, $\mathcal{M}$ is a set of m indivisible items, and $\mathbf{v} = (v_1, \ldots, v_n)$ is the utility profile with $v_i : \mathcal{M} \to \mathbb{R}$ being the utility function of agent i and $v_i(r)$ indicating i's utility for item r.

In this work, we assume that either all items are *goods* for all agents (i.e., $v_i(r) \geqslant 0$ for all $i \in \mathcal{N}$ and $r \in \mathcal{M}$), in which case we refer to I as a *goods division* instance, or all items are *chores* for all agents (i.e., $v_i(r) \leqslant 0$ for all $i \in \mathcal{N}$ and $r \in \mathcal{M}$), in which case we refer to I as a *chore division* instance.

We focus our attention to the class of additive utility functions, in which the utility of agent i for a set of items $S \subseteq \mathcal{M}$ is given by, with slight abuse of notation, $v_i(S) = \sum_{r \in S} v_i(r)$. We are interested in the following subclasses of additive utilities. Let v denote an additive utility function over a set of items $\mathcal{M}$ in a goods division or chore division instance.

Definition 2.1 (Factored utilities). We say that a utility function $v : \mathcal{M} \to \{0, p_1, \ldots, p_k\} \subset \mathbb{Z}$ is *factored* if p_j divides p_{j+1} (i.e., $p_{j+1} = q \cdot p_j$ for some $q \in \mathbb{N}_{>0}$) for each $j \in [k-1]$.

Definition 2.2 (Weakly lexicographic utilities). We say that v is *weakly lexicographic* if there is a partition $(L_1, \ldots, L_k)$ of $\mathcal{M}$ with

(1) $\forall i \in [k]$ and $r, r' \in L_i$, we have $|v(r)| = |v(r')| > 0$, and
(2) $\forall i \in [k]$ and $r \in L_i$, we have $|v(r)| > |\sum_{r' \in L_{i+1} \cup \ldots \cup L_k} v(r')|$.

Further, if $k = m$, then we say that v is (strictly) lexicographic.

Weakly lexicographic utilities can be seen as a special case of factored utilities, as we may assume that $|v_i(r)|$ is a power of m. The following lemma shows that we can make that assumption without changing the ordinal preferences over bundles. All proofs in the paper can be found in the full version [18].

Lemma 2.3. *Let v be a weakly lexicographic utility function over a set of items $\mathcal{M}$. Then, there exists a weakly lexicographic factored utility function v' given by $v' : \mathcal{M} \to \{1, m, m^2, \ldots\}$ for goods or $v' : \mathcal{M} \to \{-1, -m, -m^2, \ldots\}$ for chores such that $v(S) \leqslant v(S') \Leftrightarrow v'(S) \leqslant v'(S')$ for all $S, S' \subseteq \mathcal{M}$.*

Definition 2.4 (Bivalued utilities). We say that v is *bivalued* if there are non-zero $a, b \in \mathbb{R}$ such that $v(r) \in \{a, b\}$ for all $r \in \mathcal{M}$. In case of goods, we will use the convention $0 < a < b$, and in case of chores, we will use the convention $0 > a > b$. Further, if a divides b, we say that v is *factored bivalued.*

We say that a goods division or chore division instance has factored (resp., weakly lexicographic) utilities if every agent has a factored (resp., weakly lexicographic) utility function. The instance has bivalued utilities if all agents have bivalued utilities for some common a, b (i.e., there exist a, b such that $v_i(r) \in \{a, b\}$ for all i, r). The instance has *personalized bivalued* utilities if each agent i has a bivalued utility function (perhaps with personalized a_i, b_i).[1]

Allocations: An *allocation* $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is a collection of bundles $\mathbf{x}_i \subseteq \mathcal{M}$, one for each agent $i \in \mathcal{N}$, such that the bundles are pairwise disjoint ($\mathbf{x}_i \cap \mathbf{x}_j = \emptyset$ for all distinct $i, j \in \mathcal{N}$) and every item is allocated ($\bigcup_{i \in \mathcal{N}} \mathbf{x}_i = \mathcal{M}$).

Fairness and Efficiency Desiderata: We study two prominent fairness notions for the allocation of indivisible items, known as envy-freeness up to one item [15, 16, 33] and maximin share fairness [15, 32]. These are respectively relaxations of the classical notions of envy-freeness and of proportionality. We give definitions that work for both goods and chores [6].

Definition 2.5 (Envy-freeness up to one item). An integral allocation $\mathbf{x}$ is said to be *envy-free up to one item* (EF1) if, for every pair of agents $i, j \in \mathcal{N}$ such that $\mathbf{x}_i \cup \mathbf{x}_j \neq \emptyset$, there exists an item $r \in \mathbf{x}_i \cup \mathbf{x}_j$ such that $v_i(\mathbf{x}_i \setminus \{r\}) \geqslant v_i(\mathbf{x}_j \setminus \{r\})$.

In a goods division problem, this reduces to $v_i(\mathbf{x}_i) \geqslant v_i(\mathbf{x}_j \setminus \{g\})$ for some good $g \in \mathbf{x}_j$ (a good removed from the bundle of agent j), while in a chore division problem, it reduces to $v_i(\mathbf{x}_i \setminus \{c\}) \geqslant v_i(\mathbf{x}_j)$ for some $c \in \mathbf{x}_i$ (a chore removed from the bundle of agent i).

Definition 2.6 (Maximin share fairness). For $k \in \mathbb{N}$, let $\mathcal{P}^k(\mathcal{M})$ be the set of all partitions of $\mathcal{M}$ into k bundles. For agent $i \in \mathcal{N}$, let

$$\text{MMS}_i^k = \max_{(S_1, \ldots, S_k) \in \mathcal{P}^k(\mathcal{M})} \min_{t \in [k]} v_i(S_t).$$

Note that this is the maximum utility she can obtain by partitioning the items into k bundles and receiving the least valued bundle. We

[1]Personalized bivalued utilities are a special case of k-ary utilities [24].

refer to an optimal partition $(S_1, \ldots, S_k)$ in the above equation as a maximin k-partition for agent i. The *maximin share* of agent $i \in \mathcal{N}$ is defined as MMS_i^n. For simplicity of notation, we write MMS_i^n as MMS_i and refer to a maximin n-partition as a *maximin partition*. An allocation $\mathbf{x}$ is said to be maximin share fair (MMS) if each agent receives at least as much utility as her maximin share, i.e., if $v_i(\mathbf{x}_i) \geqslant \text{MMS}_i$ for each agent $i \in \mathcal{N}$.

Finally, we define a prominent notion of economic efficiency.

Definition 2.7 (Pareto optimality). We say that allocation $\mathbf{x}$ is *Pareto dominated* by allocation $\mathbf{x}'$ if $v_i(\mathbf{x}_i) \leqslant v_i(\mathbf{x}_i')$ for every agent $i \in \mathcal{N}$ and at least one inequality is strict. An allocation $\mathbf{x}$ is said to be *Pareto optimal* (PO) if it is not Pareto dominated by any allocation.

3 EF1 + PO FOR BIVALUED CHORES

In this section, we present a polynomial-time algorithm that finds an EF1 and PO allocation for chore division instances with bivalued utilities, thereby also establishing the existence of such allocations. Specifically, we scale agent utilities such that for some $p > 1$, the utility of each agent i for every chore c is $v_i(c) \in \{-1, -p\}$. Further, if some agent i has $v_i(c) = -p$ for all chores c, then we will scale this so that $v_i(c) = -1$ for all chores c. This will ensure that each agent values at least one chore at -1. Recall that scaling the utilities of any agent does not affect whether an allocation is EF1 or PO.

Our algorithm builds on the algorithm by Barman et al. [9] for finding an EF1 and PO allocation of goods. Their algorithm starts with a PO allocation and then moves items around until it is EF1, while maintaining that the allocation is PO at every step. Pareto optimality is maintained in the algorithm by ensuring that the allocation remains an equilibrium in a *Fisher market*. Thus, we start by introducing some basic concepts about Fisher markets.

3.1 Fisher Markets for Chore Division

A *price vector* $\mathbf{p}$ assigns a price $\mathbf{p}(c) > 0$ to each chore c. For a subset $S \subseteq \mathcal{M}$ of chores, we write $\mathbf{p}(S) = \sum_{c \in S} \mathbf{p}(c)$. Given this price vector, the *pain per buck* (PB) ratio of agent i for chore c is defined as $\text{PB}_i(c) = \frac{|v_i(c)|}{\mathbf{p}(c)}$, and the *minimum pain per buck* (MPB) ratio of agent i is defined as $\text{MPB}_i = \min_{c \in \mathcal{M}} \text{PB}_i(c)$. A chore c with $\text{PB}_i(c) = \text{MPB}_i$ is called an *MPB chore* for agent i.

Definition 3.1. A pair $(\mathbf{x}, \mathbf{p})$ of an allocation $\mathbf{x}$ and a price vector $\mathbf{p}$ is a (Fisher market) *equilibrium* if each agent is allocated only her MPB chores, i.e., if $\text{PB}_i(c) = \text{MPB}_i$ for all $i \in \mathcal{N}$ and all $c \in \mathbf{x}_i$.

We say that $\mathbf{x}$ is an equilibrium allocation if $(\mathbf{x}, \mathbf{p})$ is an equilibrium for some price vector $\mathbf{p}$. The following is known to hold by the so-called first welfare theorem.

Proposition 3.2. *Every equilibrium allocation is Pareto optimal.*

In fact, a stronger statement is true: every equilibrium allocation is *fractionally Pareto optimal* (fPO), which means it is not even Pareto dominated by a *fractional* allocation [9]. Moreover, a second welfare theorem shows that every fPO allocation arises as an equilibrium allocation [9]. For the special case of bivalued utilities, it turns out that that PO and fPO are equivalent.

Theorem 3.3. *Given a goods or chore division problem with bivalued utilities, an allocation* $\mathbf{x}$ *is Pareto optimal if and only if it is fractionally Pareto optimal.*

Thus, the stronger efficiency guarantee fPO that Fisher markets provide does not have bite in our setting. Conversely, though, this equivalence means that any method that identifies a PO allocation for bivalued utilities must implicitly calculate an equilibrium.

As an invariant, our algorithm will keep the considered allocation an equilibrium. Our aim is to find a fair equilibrium, by which we will mean that the prices of agents' bundles are approximately equal. This notion is an adaption to the chores case of a property introduced by Barman et al. [9].

Definition 3.4 (Price envy-freeness up to one item). We say that $(\mathbf{x}, \mathbf{p})$ is *price envy-free up to one item* (pEF1) if, for all $i, j \in \mathcal{N}$ with $\mathbf{x}_i \neq \emptyset$, there is a chore $c \in \mathbf{x}_i$ such that $\mathbf{p}(\mathbf{x}_i \setminus \{c\}) \leqslant \mathbf{p}(\mathbf{x}_j)$.

Like for the goods division case [9], pEF1 implies EF1.

Lemma 3.5. *If* $(\mathbf{x}, \mathbf{p})$ *is a pEF1 equilibrium, then* $\mathbf{x}$ *is EF1.*

For $S \subseteq \mathcal{M}$, define $\mathbf{p}_{\text{up to 1}}(S) = \mathbf{p}(S) - \max_{c \in S} \mathbf{p}(c)$, if $S \neq \emptyset$, and 0 if $S = \emptyset$. We often write $\text{ls} \in \mathcal{N}$ for the *least spender*, i.e., an agent $\text{ls} \in \arg\min_{i \in \mathcal{N}} \mathbf{p}(\mathbf{x}_i)$. Then we see that $(\mathbf{x}, \mathbf{p})$ is pEF1 if and only if $\mathbf{p}_{\text{up to 1}}(\mathbf{x}_i) \leqslant \mathbf{p}(\mathbf{x}_{\text{ls}})$ for all $i \in \mathcal{N}$. Let us call an agent $i \in \mathcal{N}$ a *violator* if $\mathbf{p}_{\text{up to 1}}(\mathbf{x}_i) > \mathbf{p}(\mathbf{x}_{\text{ls}})$. Thus, $(\mathbf{x}, \mathbf{p})$ is pEF1 if and only if no agent is a violator.

Given an equilibrium $(\mathbf{x}, \mathbf{p})$, we write $j \xleftarrow{c} i$ if agent i owns item c (so $c \in \mathbf{x}_i$) and c is an MPB chore for j. Thus, if we have $j \xleftarrow{c} i$ then the allocation $\mathbf{x}'$ obtained from $\mathbf{x}$ by transferring item c from i to j is still an equilibrium.

Definition 3.6 (MPB alternating path). An *MPB alternating path* of length ℓ from i_ℓ to i_0 is a sequence $i_0 \xleftarrow{c_1} i_1 \xleftarrow{c_2} \cdots \xleftarrow{c_\ell} i_\ell$.

If there exists an MPB alternating path from i_ℓ to i_0, we write $i_0 \leftsquigarrow i_\ell$. We always have $i_0 \leftsquigarrow i_0$.

3.2 Algorithm

We now present Algorithm 1 which computes an PO and EF1 allocation given a chore division instance with bivalued utilities.

Theorem 3.7. *Given a chore division problem* $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$ *with bivalued utilities, Algorithm 1 finds a PO and EF1 allocation in* $\text{poly}(n, m)$ *time.*

The algorithm starts with an $(\mathbf{x}, \mathbf{p})$ that is guaranteed to be an equilibrium. Then, it proceeds in *iterations*. The value k, maintained by the algorithm, signifies the current iteration number. In each iteration k, the algorithm goes through Phases 2a, 2b, and 3 (except that in the final iteration the algorithm terminates after Phase 2b). During Phases 2a and 2b, the algorithm keeps the price vector $\mathbf{p}$ fixed and updates the allocation $\mathbf{x}$, and in the subsequent Phase 3, it then keeps the allocation $\mathbf{x}$ fixed, identifies a certain set H_k of agents and updates the price vector $\mathbf{p}$ by reducing the prices of the chores allocated to H_k by a multiplicative factor α.

A key property of our algorithm is that it ensures that the sets H_k are disjoint across different iterations. This helps prove that our algorithm always terminates after at most n iterations, since each H_k contains at least one agent. This property differentiates our algorithm from the algorithm of Barman et al. [9] for allocating goods and requires us to introduce Phase 2a, which is not present in their algorithm. Phase 2b, on the other hand, is very similar to Phase 2 in their algorithm.

ALGORITHM 1: EF1 + PO for Bivalued Chores

```
1  Phase 1 Initialization
2      Let $\mathbf{x}$ be an allocation maximizing social welfare $\sum_{i\in\mathcal{N}} v_i(\mathbf{x}_i)$.
3      For each $c \in \mathcal{M}$, let $\mathbf{p}_c = p \cdot |\max_{i\in\mathcal{N}} v_i(c)|$
4      $k \leftarrow 1$, the number of the current iteration
5  Phase 2a Reallocate chores
6      for $\ell \in (k-2, k-3, \ldots, 2, 1)$ do
7          while true do
8              $i \leftarrow$ an agent from $\arg\max_{i\in H_\ell} \mathbf{p}_{\text{up to }1}(\mathbf{x}_i)$
9              $j \leftarrow$ an agent from $\arg\min_{j\in H_{\ell+1}\cup\cdots\cup H_{k-1}} \mathbf{p}(\mathbf{x}_j)$
10             if $\mathbf{p}_{\text{up to }1}(\mathbf{x}_i) > \mathbf{p}(\mathbf{x}_j)$ then
11                 $c \leftarrow$ any item from $\mathbf{x}_i \setminus \text{entitled}(i)$
12                 Transfer $c$ from $i$ to $j$
13             else
14                 break
15 Phase 2b Reallocate chores
16     while true do
17         ls $\leftarrow$ an agent from $\arg\min_{i\in\mathcal{N}} \mathbf{p}(\mathbf{x}_i)$
18         if there is an MPB alternating path ls $\overset{c_1}{\leftarrow} i_1 \overset{c_2}{\leftarrow} \cdots \overset{c_\ell}{\leftarrow} i_\ell$ with
             $\mathbf{p}_{\text{up to }1}(\mathbf{x}_{i_\ell}) > \mathbf{p}(\mathbf{x}_{\text{ls}})$ then
19             Choose such a path of minimum length $\ell$
20             Transfer $c_\ell$ from $i_\ell$ to $i_{\ell-1}$
21         else
22             break
23     if $\mathbf{x}$ satisfies pEF1 then
24         return $\mathbf{x}$
25 Phase 3 Price reduction
26     $H_k \leftarrow \{i \in \mathcal{N}$ :
         there is an agent ls $\in \arg\min_{i\in\mathcal{N}} \mathbf{p}(\mathbf{x}_i)$ with ls $\leftrightsquigarrow i\}$
27     ► Timestamp: $t_{k,b}$
28     $\alpha \leftarrow \min\{\text{PB}_i(c)/\text{MPB}_i : i \in H_k, c \in \bigcup_{j\in\mathcal{N}\setminus H_k} \mathbf{x}_j\}$
29     for $i \in H_k$ do
30         entitled$(i) \leftarrow \mathbf{x}_i$
31         for $c \in \mathbf{x}_i$ do
32             $\mathbf{p}_c \leftarrow \frac{1}{\alpha} \cdot \mathbf{p}_c$
33     ► Timestamp: $t_{k,a}$
34     $k \leftarrow k + 1$
35     Start Phase 2a (i.e. go to line 5)
```

Another key ingredient of our algorithm is that once an agent i is assigned to a set H_k, the chores assigned to i at that time become *entitled chores* of agent i, denoted entitled(i). These are the chores which went through a price reduction while they were allocated to agent i. Subsequently the algorithm will never move the entitled chores away from i. Finally, in order to reason about the equilibria at different times during the execution of the algorithm, we timestamp important steps of the algorithm: $t_{k,b}$ and $t_{k,a}$ denote the time right *before* and right *after* the execution of Phase 3 in iteration k.

We prove the correctness of the algorithm by induction on k. We claim that for all $k \geqslant 1$ such that Algorithm 1 reaches time $t_{k,a}$,

(H1) $H_k \cap H_\ell = \emptyset$ for all $1 \leqslant \ell < k$.
(H2) During iteration k, each time the algorithm reaches line 11, there exists a chore $c \in \mathbf{x}_i \setminus \text{entitled}(i)$. All such chores are MPB chores for agent j.
(H3) At time $t_{k,b}$, each $i \in H_1 \cup \cdots \cup H_k$ is not a violator, so $\mathbf{p}_{\text{up to }1}(\mathbf{x}_i) \leqslant \mathbf{p}(\mathbf{x}_{\text{ls}})$ where ls is the least spender.
(H4) At time $t_{k,a}$, each $i \in H_1 \cup \cdots \cup H_k$ owns every entitled item, entitled$(i) \subseteq \mathbf{x}_i$.
(H5) When line 28 is reached during iteration k, α is set to p.
(H6) At time $t_{k,a}$, we have $\mathbf{p}(c) \in \{1, p\}$ for all $c \in \mathcal{M}$. If $\mathbf{p}(c) = 1$, then $c \in \text{entitled}(i)$ for some $i \in H_1 \cup \cdots \cup H_k$.
(H7) At time $t_{k,a}$, we have $\text{MPB}_i = 1$ for all $i \in H_1 \cup \cdots \cup H_k$, and $\text{MPB}_i = 1/p$ for all other agents.

Let us first check that these statements together imply that $(\mathbf{x}, \mathbf{p})$ remains an equilibrium throughout the execution of the algorithm, and that the algorithm terminates in polynomial time, in line 24. Then $(\mathbf{x}, \mathbf{p})$ is an equilibrium satisfying pEF1, and thus we have found an PO and EF1 allocation, as required.

Lemma 3.8. *Assume that (H1) to (H7) hold for all $k \geqslant 1$ such that Algorithm 1 reaches time $t_{k,a}$. Then, throughout the algorithm's execution, $(\mathbf{x}, \mathbf{p})$ is an equilibrium.*

For the statement about termination, we need a few properties of Phase 2b of the algorithm, which is very similar to Phase 2 of the original algorithm due to Barman et al. [9].

Lemma 3.9 (Properties of Phase 2b). *Consider a run of Phase 2b, and assume that $(\mathbf{x}, \mathbf{p})$ is an equilibrium at the start of the run.*

(1) The run terminates after poly(n, m) time.
(2) Least spending $\min_{i\in\mathcal{N}} \mathbf{p}(\mathbf{x}_i)$ never decreases during the run.

Assuming the induction hypotheses and using the lemmas mentioned above, we can now prove that the algorithm terminates, and is hence correct.

Lemma 3.10. *Assume that (H1) to (H7) hold for all $k \geqslant 1$ such that Algorithm 1 reaches time $t_{k,a}$. Then the algorithm terminates in polynomial time and returns a pEF1 equilibrium.*

We now turn to proving our induction hypotheses. Recall that we prove them by induction on the iteration number k. First, let us prove them in the base of $k = 1$.

Lemma 3.11 (Base case). *(H1) to (H7) hold for $k = 1$.*

From now on, we assume that (H1) to (H7) hold for all ℓ with $1 \leqslant \ell \leqslant k$ for some $k \geqslant 1$. Our goal is to show that (H1) to (H7) hold for iteration $k + 1$.

The next lemma proves the usefulness of Phase 2a, which is that at the end of this phase, no agent in $H_1 \cup \cdots \cup H_k$ (i.e., no agent who has gone through a price reduction) can be a violator.

Lemma 3.12. *Let t_{mid} denote the time when the algorithm reaches line 16 in iteration $k + 1$, i.e. when Phase 2a ends and Phase 2b begins. We claim that at time t_{mid}, no agent in $H_1 \cup \cdots \cup H_k$ is a violator.*

The next lemma proves a useful guarantee for Phase 2b.

Lemma 3.13. *During the execution of Phase 2b in iteration $k + 1$, no entitled items are transferred. Further, at the end of Phase 2b, no agent $i \in H_1 \cup \cdots \cup H_k$ is a violator.*

Finally, we prove the induction step of our induction hypotheses.

Lemma 3.14. *Suppose Algorithm 1 reaches time $t_{k+1,a}$. Then (H1) to (H7) hold for $k + 1$.*

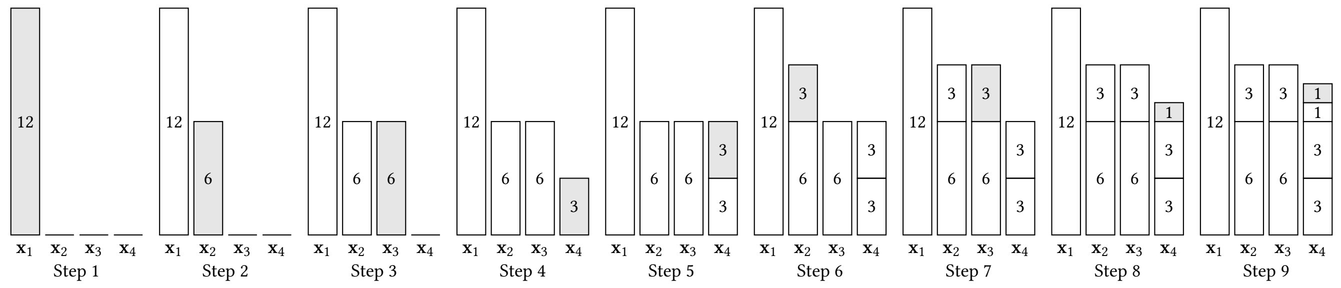

Figure 2: An execution of Algorithm 2 to obtain a maximin 4-partition for the factored utility function $v = (12, 6, 6, 3, 3, 3, 3, 1, 1)$. For each step, the gray box indicates the item placed by the algorithm in that step.

4 MMS UNDER RESTRICTED UTILITIES

As discussed earlier, MMS allocations are not guaranteed to exist for arbitrary additive utilities. Prior work on allocating goods establishes that they always exist for binary utilities [13] and strictly lexicographic utilities [29]. We generalize these results to the classes of weakly lexicographic and factored personalized bivalued utilities. The following theorem summarizes our main result of this section.

Theorem 4.1. *In every goods or chore division instance with weakly lexicographic or factored personalized bivalued utilities, an MMS allocation always exists and can be computed in polynomial time.*

4.1 Ordered Instances and Valid Reductions

Let us begin by reviewing two basic techniques which are commonly used in the literature on computing MMS allocations. Throughout this section, we let $\mathcal{N} = [n]$ and $\mathcal{M} = [m]$.

4.1.1 Ordered Instances. Bouveret and Lemaître [13, Prop. 14] show that when dealing with MMS allocations, one can assume, without loss of generality, that all agents have the same preference ranking over the items. This result was originally stated for goods, but the same proof works for chores as well.

Lemma 4.2 ([13]). *Let $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$ be a goods or chore division instance. Let $I' = (\mathcal{N}, \mathcal{M}, \mathbf{v}')$ be an* ordered instance *where, for each $i \in \mathcal{N}$, v'_i is a permutation of v_i such that $|v'_i(1)| \geqslant \ldots \geqslant |v'_i(m)|$. If $\mathbf{x}'$ is an MMS allocation for I', then there exists an MMS allocation $\mathbf{x}$ for I. Given $\mathbf{x}'$, one can compute $\mathbf{x}$ in polynomial time.*

Given Lemma 4.2, we will assume that all instances in this section (except in Section 4.5, where our goal is to achieve PO in conjunction with MMS) are ordered instances. Specifically, we will assume that $|v_i(1)| \geqslant \ldots \geqslant |v_i(m)|$ for each agent $i \in \mathcal{N}$.

4.1.2 Valid Reductions. Another common idea used in the literature on finding (approximate) MMS allocations is that of *valid reductions* [3, 22, 26, 27, 31, 32].

Definition 4.3 (Valid Reduction). Let $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$ be a goods or chore division instance, $i \in \mathcal{N}$ be an agent, and $S \subseteq \mathcal{M}$ be a subset of items. The pair (i, S) is a *valid reduction* if

(1) $v_i(S) \geqslant \text{MMS}_i^n(\mathcal{M})$, and
(2) $\text{MMS}_j^{n-1}(\mathcal{M} \setminus S) \geqslant \text{MMS}_j^n(\mathcal{M})$ for all $j \in \mathcal{N} \setminus \{i\}$.

If (i, S) is a valid reduction, we can allocate bundle S to agent i, and ignore i and S subsequently. Formally, consider the reduced instance $I' = (\mathcal{N} \setminus \{i\}, \mathcal{M} \setminus S, \mathbf{v})$ obtained from I by removing i and S. Then if $\mathbf{x}'$ is an MMS allocation for I', then the allocation $\mathbf{x}$ with $\mathbf{x}_i = S$ and $\mathbf{x}_j = \mathbf{x}'_j$ for all $j \neq i$ is an MMS allocation for I. This holds because agent i receives her MMS value in $\mathbf{x}$ by (1), and for any other agent j, $v_j(\mathbf{x}'_j) \geqslant \text{MMS}_j^{n-1}(\mathcal{M} \setminus S) \geqslant \text{MMS}_j^n(\mathcal{M})$ by (2).

Our proofs for both goods and chore division under both weakly lexicographic and factored personalized bivalued utilities work in the same fashion: we show that every instance admits a valid reduction which can be computed efficiently. The next lemma identifies one of the ways of finding a valid reduction.

Lemma 4.4. *For a goods or chore division instance $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$, the pair (i, S), where $i \in \mathcal{N}$ and $S \subseteq \mathcal{M}$ is a valid reduction if $v_i(S) \geqslant \text{MMS}_i^n(\mathcal{M})$ and, for all agents $i' \in \mathcal{N} \setminus \{i\}$, there is a maximin n-partition $P_{i'} = (S'_1, \ldots, S'_n)$ of agent i' and a bundle $S' \in P_{i'}$ with $S \subseteq S'$ for goods division and $S \supseteq S'$ for chore division.*

4.2 Exact MMS Value for Factored Utilities

Note that in order to check the validity of a reduction (i, S) via Lemma 4.4, we need to relate S to one of the bundles in a maximin n-partition of every agent other than i. For this, we need to reason about what a maximin n-partition looks like for an agent. We show that in any goods or chore division instance with factored utilities (which covers weakly lexicographic and personalized factored bivalued utilities as special cases), a maximin n-partition of an agent (and hence, her MMS value) can be computed efficiently. This is in sharp contrast to the case of general additive utilities, for which the problem is known to be NP-hard for both goods and chores [21].

Algorithm 2 works for both goods division and chore division. It considers items in nonincreasing order of their absolute value and greedily assigns them to a bundle with lowest total absolute value. Figure 2 shows the execution of the algorithm on an example.

ALGORITHM 2: Compute a maximin n-partition for a factored utility function v

```
1 $\mathbf{x} \leftarrow (\mathbf{x}_i = \emptyset)_{i \in \mathcal{N}}$            // x denotes a partial allocation
2 for $r \in \mathcal{M}$ in a nonincreasing order of $|v(r)|$ do
3     $k^* \leftarrow \arg\min_{k \in \mathcal{N}} |v(\mathbf{x}_k)|$
4     $\mathbf{x}_{k^*} \leftarrow \mathbf{x}_{k^*} \cup \{r\}$
5 return $\mathbf{x}$
```

Lemma 4.5. *Given a factored utility function v over a set of items $\mathcal{M}$ (all goods or all chores), Algorithm 2 efficiently computes a maximin n-partition of $\mathcal{M}$ under v.*

When we assume our instance to be ordered, we will consider the items in the standard order $1, \dots, m$ in Algorithm 2. This will allow us to reason about the exact indices of items allocated to different bundles under Algorithm 2.

Algorithm 2 does not always work for utility functions v that are not factored. For example, let $v = (3, 3, 2, 2, 2)$ and $n = 2$. The algorithm produces the partition $\mathbf{x}_1 = \{3, 2, 2\}$ and $\mathbf{x}_2 = \{3, 2\}$, for a minimum share of 5. However, the unique MMS partition is $\mathbf{x}_1 = \{3, 3\}$ and $\mathbf{x}_2 = \{2, 2, 2\}$, which achieves a maximin share of 6.

4.3 Weakly Lexicographic Utilities

We now present a valid reduction for weakly lexicographic utilities. First, we introduce the concept of a "bad cut". Recall that we work with ordered instances in which $|v_i(1)| \geqslant \dots \geqslant |v_i(m)|$ for all i.

Definition 4.6 (Bad Cuts). In a goods or chore division instance $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$, we say that index $k \in [m-1]$ is a *cut* of agent i if $v_i(k) \neq v_i(k+1)$. Further, if k is not a multiple of n, we say that it is a *bad cut* of agent i. Define C_i to be the smallest bad cut of agent i; let $C_i = m$ if agent i does not have any bad cuts.

i	$\lvert v_1\rvert$	$\lvert v_2\rvert$	$\lvert v_3\rvert$	$\lvert v_4\rvert$	$\lvert v_5\rvert$	$\lvert v_6\rvert$	$\lvert v_7\rvert$	$\lvert v_8\rvert$	$\lvert v_9\rvert$	C_i
i_1	81	81	81	$81\rvert_4$	9	9	$9\rvert_7$	1	1	4
i_2	81	81	$81\rvert_{\checkmark}$	9	9	$9\rvert_{\checkmark}$	1	1	1	9
i_3	$729\rvert_1$	81	81	$81\rvert_4$	9	9	$9\rvert_7$	1	1	1

Table 1: An instance with weakly lexicographic utilities. Bad cuts at index k are shown as $|_k$, and non-bad cuts as $|_{\checkmark}$.

Example 4.7. The ordered instance described in Table 1 consists of $n = 3$ agents and $m = 9$ items. The *cuts* of i_1, i_2, and i_3 are $\{4, 7\}$, $\{3, 6\}$, and $\{1, 4, 7\}$ respectively. Here, a cut is considered a *bad cut* if it is not divisible by $n = 3$. Then, all cuts of i_1 and i_3 are bad while i_2 has no bad cuts. Following the definition of C_i's, we have $C_{i_1} = 4$, $C_{i_2} = m = 9$, and $C_{i_3} = 1$.

First, for any agent i, we identify a specific bundle in a maximin n-partition of agent i produced by Algorithm 2, in terms of C_i.

Lemma 4.8. *For a goods or chore division instance $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$ with weakly lexicographic utilities and agent $i \in \mathcal{N}$, there exists a maximin n-partition of agent i in which one of the bundles is $S = \{1, n+1, \dots, kn+1\}$, where $k = \lfloor (C_i - 1)/n \rfloor$.*

Lemma 4.8 shows that in Example 4.7 there exists a maximin 3 partition for i_1 where one of the bundles is $\{1, 4\}$. Same applies to $\{1, 4, 7\}$ and $\{1\}$ for i_1, i_2 and i_3 respectively. Following this observation and assuming items here are all goods, allocating $\{1\}$ to i_3 is a valid reduction by Lemma 4.4, because $\{1\}$ is a subset of the other two bundles described. If the items were chores, i.e. values being costs, allocating $\{1, 4, 7\}$ to i_2 would have formed a valid reduction.

Next, we show that if we choose an agent i with the minimum or maximum C_i and the corresponding S from Lemma 4.8, the pair (i, S) forms a valid reduction. Note that this valid reduction can be found in polynomial time.

Lemma 4.9. *For a goods (respectively, chore) division instance $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$ with weakly lexicographic utilities, the pair (i, S) is a valid reduction when i is an agent with the minimum (resp., maximum) C_i and $S = \{1, n+1, \dots, kn+1\}$, where $k = \lfloor (C_i - 1)/n \rfloor$.*

4.4 Factored Personalized Bivalued Utilities

In this section, we present a valid reduction for factored personalized bivalued utilities. Recall that we work with ordered instances. Hence, for each agent i, there exists $k \in [m]$ such that $|v_i(r)| = p_i$ for all $r \leqslant k$ and $|v_i(r)| = 1$ for all $r > k$. Thus, each agent i has at most one cut (k, if $k < m$), and C_i is equal to this cut (if it exists and it is bad) and m otherwise. However, in this case, simply choosing an agent i with the minimum or maximum C_i does not work. Instead, we rely on a different metric, called "idle time".

Definition 4.10 (Idle Time). In a goods or chore division instance $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$ with factored personalized bivalued utilities, we define AC_i as 0 if $C_i = m$ and as $n - (C_i \bmod n)$ otherwise. Let the *idle time* of agent i to be $T_i^{\text{idle}} = \min\{p_i \cdot AC_i, m - C_i\}$.

i	$\lvert v_1\rvert$	$\lvert v_2\rvert$	$\lvert v_3\rvert$	$\lvert v_4\rvert$	$\lvert v_5\rvert$	$\lvert v_6\rvert$	$\lvert v_7\rvert$	$\lvert v_8\rvert$	$\lvert v_9\rvert$	C_i	AC_i	T_i^{idle}
i_1	2	2	2	$2\ \rvert_4$	$\underline{\mathbf{1}}$	$\underline{\mathbf{1}}$	$\underline{1}$	$\underline{1}$	1	4	2	4
i_2	$5\ \rvert_1$	$\underline{\mathbf{1}}$	$\underline{\mathbf{1}}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	1	2	8
i_3	4	4	4	4	4	4	4	$4\ \rvert_8$	$\underline{\mathbf{1}}$	8	1	1

Table 2: An instance with personalized bivalued utilities. Bold cells refer to active bundles AC_i, and the underlined ones refer to idle times T_i^{idle}.

Example 4.11. Table 2 presents an ordered instance with personalized bivalued utilities where $p_{i_1} = 2$, $p_{i_2} = 5$ and $p_{i_3} = 4$. The number of *active bundles* and *idle times* are also shown in the table.

First, note that when agent i does not admit a bad cut, we have $C_i = m$, $AC_i = 0$, and $T_i^{\text{idle}} = 0$. Suppose agent i admits a bad cut $C_i = kn + r$ with remainder $r \in [n-1]$. Observe that Algorithm 2 operates in at most three phases. In the first phase, it divides the items with absolute value p_i in a round robin fashion between all n bundles, until it reaches the bad cut. At that time, $C_i \bmod n$ bundles have an extra item with absolute value p_i. We refer to the remaining bundles as the *"active bundles"*; note that there are precisely AC_i many active bundles. In the second phase, it divides the items with absolute value 1 between the active bundles in a round robin fashion, until either all items are allocated or all n bundles become of exactly equal value (this is where the assumption of the utilities being factored, i.e., p_i being an integer is crucial). Note that the duration of this second phase is precisely the idle time of agent i defined above. If there are any remaining items with absolute value 1, the algorithm divides them between all n bundles in a round robin fashion in the final phase.

Using this observation, we are ready to characterize one of the bundles in some maximin n-partition of agent i.

Lemma 4.12. *For a goods or chore division instance $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$ with factored personalized bivalued utilities and agent $i \in \mathcal{N}$, there exists a maximin n-partition of agent i in which one of the bundles is $S = \{1, n+1, \dots, kn+1\}$, where $k = \lfloor (m - \max\{T_i^{\text{idle}} - AC_i, 0\} - 1)/n \rfloor$.*

In Example 4.11, we can find a valid reduction similar to the case of weakly lexicographic utilities. By Lemma 4.12, there is a maximin 3 partition for i_1 where one of the bundles is $\{1, 4, 7\}$. Same holds for bundles $\{1\}$ and $\{1, 4, 7\}$ for i_2 and i_3 respectively. If items were goods, the pair of i_2 with $\{1\}$ would have been a valid reduction, and the pair i_1 and $\{1, 4, 7\}$ would have worked if they were chores.

Now, we can show that choosing agent i with the minimum or maximum $\max\{(T_i^{\text{idle}} - AC_i), 0\}$ and the corresponding S from Lemma 4.12 yields a valid reduction (i, S).

Lemma 4.13. *For a goods (respectively, chore) division instance $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$ with weakly lexicographic utilities, the pair (i, S) is a valid reduction when i is an agent with the maximum (resp., minimum) value of $\max\{(T_i^{\text{idle}} - AC_i), 0\}$ and $S = \{1, n+1, \ldots, kn+1\}$, where $k = \lfloor (m - \max\{(T_i^{\text{idle}} - AC_i), 0\} - 1)/n \rfloor$.*

4.5 Achieving Pareto Optimal MMS Allocations

In this section, we show that for weakly lexicographic as well as for factored bivalued instances, we can compute an allocation that is MMS and PO in polynomial time. Our approach uses the fact that if $\mathbf{x}$ is an MMS allocation, and $\mathbf{x}'$ is a Pareto improvement over $\mathbf{x}$ then $\mathbf{x}'$ is also MMS. Thus to find an MMS and PO allocation, we can compute an MMS allocation using Theorem 4.1 and then repeatedly find Pareto improvements until we reach a PO allocation. In this section, we will show that we can in polynomial time find Pareto improvements if they exist, and that we will reach a PO allocation after at most polynomially many Pareto improvements.

Aziz et al. [4] prove that in case of goods division with weakly lexicographic or bivalued utilities, one can efficiently test if a given allocation is Pareto optimal (PO). Further, if it is not PO, a Pareto dominating allocation with special properties always exists and can be computed efficiently. The following lemma states their result for goods division, together with an extension to chore division. While in the case of weakly lexicographic utilities our proof for chores almost mirrors their proof for goods, the ideas needed in the case of bivalued utilities are slightly different for chores. Also, the statement below is their claim for weakly lexicographic utilities; while they make a differently worded claim for bivalued utilities, their proof also shows that this claim holds for bivalued utilities.

Lemma 4.14. *In a goods or chore division instance with weakly lexicographic or bivalued utilities, one can efficiently test whether a given allocation $\mathbf{x}$ is Pareto optimal. Further, if $\mathbf{x}$ is not Pareto optimal, then there exists a cycle of distinct agents $(i_1, \ldots, i_k, i_{k+1} = i_1)$ and a cycle of distinct items $(r_1, \ldots, r_k, r_{k+1} = r_1)$ such that:*

1. *$r_t \in \mathbf{x}_{i_t}$ and $v_{i_t}(r_{t-1}) \geqslant v_{i_t}(r_t)$ for each $t \in \{2, \ldots, k+1\}$,*
2. *at least one of the above inequalities is strict, and*
3. *the allocation $\mathbf{x}^*$ obtained from $\mathbf{x}$ by reallocating item r_{t-1} to agent i_t for $t \in \{2, \ldots, k+1\}$ is a Pareto improvement over $\mathbf{x}$.*

Such a Pareto improvement $\mathbf{x}^*$ can be computed in polynomial time.

For bivalued instances, the conclusion of Lemma 4.14 also follows from our Theorem 3.3 which shows that for bivalued utilities Pareto optimality and fractional Pareto optimality are equivalent, together with the fact that fractional Pareto optimality can be checked in polynomial time via linear programming. In fact, the proofs of Lemma 4.14 and Theorem 3.3 are very similar.

The next lemma shows that starting from any allocation, if we repeatedly find a Pareto improvement using Lemma 4.14, then we obtain a Pareto optimal allocation in a polynomial number of steps.

Lemma 4.15. *Let $\mathbf{x}^0$ be an allocation in a goods division or chore division instance with weakly lexicographic or bivalued utilities. Let $(\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \ldots)$ be a chain in which, for each $k \geqslant 1$, $\mathbf{x}^k$ is a Pareto improvement over $\mathbf{x}^{k-1}$ satisfying the properties in Lemma 4.14. Then, the chain terminates at a Pareto optimal allocation in at most a polynomial number of steps.*

In Lemma 4.15, note that if the initial allocation $\mathbf{x}$ is an MMS allocation, then the final allocation must be both MMS and PO, since Pareto improvements preserve the MMS property. Plugging in the MMS allocation obtained in Theorem 4.1 as the initial allocation, we obtain the following result.

Corollary 4.16. *In every goods division or chore division instance with weakly lexicographic or factored bivalued utilities, an MMS and PO allocation always exists and can be computed in polynomial time.*

5 DISCUSSION

We make progress on the open question regarding the existence of an envy-free up to one item (EF1) and Pareto optimal (PO) allocation of chores, by giving a positive answer for the special case when agents have bivalued utilities (i.e., all utilities are in $\{a, b\}$ for some $0 > a > b$). Our algorithm uses the Fisher market framework, which has been used successfully for allocating goods [9], but requires novel ideas to adapt it to allocate chores. In case of goods with bivalued utilities, Amanatidis et al. [2] show that an allocation satisfying the stronger fairness guarantee of envy-freeness up to any good (EFX) always exists and can be computed efficiently; they also establish the existence of an EFX + PO allocation. Garg and Murhekar [23] improve upon this by using the Fisher market framework to compute an EFX + PO allocation efficiently. Investigating whether EFX or EFX + PO allocations of *chores* always exist with bivalued utilities and, if so, whether they can be computed efficiently is an exciting future direction. Alternatively, establishing the existence (and efficient computation) of EF1 + PO allocations of chores under other natural classes of utility functions, such as weakly lexicographic utilities, is also an appealing avenue for future work. Yet another direction would be to adapt our algorithm to achieve EF1 + PO allocations of *mixed items* (where some are goods but others are chores), at least under restricted utilities.

Regarding our results on maximin share fairness (MMS), recall that the existence of an MMS allocation immediately implies the existence of an MMS + PO allocation because Pareto improvements preserve the MMS guarantee. However, computing an MMS + PO allocation may not always be easy, even when computing an MMS allocation is. To the best of our knowledge, our result is the first to establish non-trivial efficient computation of an MMS + PO allocation under a natural class of utility functions. It would be interesting to try to achieve MMS for more general classes of utility functions, such as general bivalued utilities (when b/a is not an integer) or all factored valuations.

ACKNOWLEDGMENTS

Nisarg Shah was partially supported by an NSERC Discovery Grant.

REFERENCES

[1] Hannaneh Akrami, Bhaskar Ray Chaudhury, Kurt Mehlhorn, Golnoosh Shahkarami, and Quentin Vermande. 2021. Maximizing Nash Social Welfare in 2-Value Instances. arXiv:2107.08965.

[2] Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, Alexandros Hollender, and Alexandros A Voudouris. 2021. Maximum Nash welfare and other stories about EFX. *Theoretical Computer Science* 863 (2021), 69–85.

[3] Georgios Amanatidis, Evangelos Markakis, Afshin Nikzad, and Amin Saberi. 2017. Approximation algorithms for computing maximin share allocations. *ACM Transactions on Algorithms (TALG)* 13, 4 (2017), 1–28.

[4] Haris Aziz, Péter Biró, Jérôme Lang, Julien Lesca, and Jérôme Monnot. 2019. Efficient reallocation under additive and responsive preferences. *Theoretical Computer Science* 790 (2019), 1–15.

[5] Haris Aziz and Ethan Brown. 2020. Random Assignment Under Bi-Valued Utilities: Analyzing Hylland-Zeckhauser, Nash-Bargaining, and other Rules. arXiv:2006.15747.

[6] Haris Aziz, Ioannis Caragiannis, Ayumi Igarashi, and Toby Walsh. 2022. Fair allocation of indivisible goods and chores. *Autonomous Agents and Multi-Agent Systems* 36, 3 (2022), 1–21.

[7] Haris Aziz, Gerhard Rauchecker, Guido Schryen, and Toby Walsh. 2017. Algorithms for max-min share fair allocation of indivisible chores. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*. 335–341.

[8] Siddharth Barman and Sanath Kumar Krishnamurthy. 2019. On the proximity of markets with integral equilibria. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*. 1748–1755.

[9] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. 2018. Finding fair and efficient allocations. In *Proceedings of the 2018 ACM Conference on Economics and Computation (EC)*. 557–574.

[10] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. 2018. Greedy Algorithms for Maximizing Nash Social Welfare. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 7–13.

[11] Anna Bogomolnaia, Hervé Moulin, Fedor Sandomirskiy, and Elena Yanovskaya. 2017. Competitive division of a mixed manna. *Econometrica* 85, 6 (2017), 1847–1871.

[12] Shant Boodaghians, Bhaskar Ray Chaudhury, and Ruta Mehta. 2022. Polynomial Time Algorithms to Find an Approximate Competitive Equilibrium for Chores. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2285–2302.

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

[14] Simina Brânzei and Fedor Sandomirskiy. 2019. Algorithms for competitive division of chores. arXiv:1907.01766.

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

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

[17] Nikhil R Devanur, Christos H Papadimitriou, Amin Saberi, and Vijay V Vazirani. 2008. Market equilibrium via a primal–dual algorithm for a convex program. *Journal of the ACM (JACM)* 55, 5 (2008), 1–18.

[18] Soroush Ebadian, Dominik Peters, and Nisarg Shah. 2021. How to Fairly Allocate Easy and Difficult Chores. arXiv:2110.11285.

[19] Duncan Karl Foley. 1967. *Resource allocation and the public sector.* Yale University.

[20] George Gamow and Marvin Stern. 1958. *Puzzle-Math.* Viking.

[21] Michael R. Garey and David S. Johnson. 1990. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co.

[22] Jugal Garg, Peter McGlaughlin, and Setareh Taki. 2019. Approximating Maximin Share Allocations. In *Proceedings of the 2nd Symposium on Simplicity in Algorithms (SOSA)*, Vol. 69. 20:1–20:11.

[23] Jugal Garg and Aniket Murhekar. 2021. Computing Fair and Efficient Allocations with Few Utility Values. In *International Symposium on Algorithmic Game Theory*. 345–359.

[24] Jugal Garg and Aniket Murhekar. 2021. On Fair and Efficient Allocations of Indivisible Goods. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*. 5595–5602.

[25] Jugal Garg, Aniket Murhekar, and John Qin. 2022. Fair and Efficient Allocations of Chores under Bivalued Preferences. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*.

[26] Jugal Garg and Setareh Taki. 2021. An improved approximation algorithm for maximin shares. *Artificial Intelligence* 300 (2021), 103547.

[27] Mohammad Ghodsi, MohammadTaghi HajiAghayi, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. 2018. Fair allocation of indivisible goods: Improvements and generalizations. In *Proceedings of the 19th ACM Conference on Economics and Computation (EC)*. 539–556.

[28] Daniel Halpern, Ariel D Procaccia, Alexandros Psomas, and Nisarg Shah. 2020. Fair division with binary valuations: One rule to rule them all. In *Proceedings of the 16th International Conference on Web and Internet Economics (WINE)*. Springer, 370–383.

[29] Hadi Hosseini, Sujoy Sikdar, Rohit Vaish, and Lirong Xia. 2021. Fair and Efficient Allocations under Lexicographic Preferences. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*. 5472–5480.

[30] Xin Huang and Pinyan Lu. 2021. An algorithmic framework for approximating maximin share allocation of chores. In *Proceedings of the 22nd ACM Conference on Economics and Computation (EC)*. 630–631.

[31] David Kurokawa, Ariel D Procaccia, and Junxing Wang. 2016. When can the maximin share guarantee be guaranteed?. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*. 523–529.

[32] David Kurokawa, Ariel D Procaccia, and Junxing Wang. 2018. Fair enough: Guaranteeing approximate maximin shares. *Journal of the ACM (JACM)* 65, 2 (2018), 1–27.

[33] Richard J Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. 2004. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Economics and Computation (EC)*. 125–131.

[34] James B Orlin. 2010. Improved algorithms for computing Fisher's market clearing prices. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC)*. 291–300.

[35] Nisarg Shah. 2017. Spliddit: two years of making the world fairer. *XRDS: Crossroads, The ACM Magazine for Students* 24, 1 (2017), 24–28.

[36] Hal R. Varian. 1974. Equity, envy and efficiency. *Journal of Economic Theory* 9 (1974), 63–91.