How to Fairly Allocate Easy and Difficult Chores

SOROUSH EBADIAN, University of Toronto
DOMINIK PETERS, University of Toronto
NISARG SHAH, University of Toronto

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 one that is both PO and MMS 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.

1 INTRODUCTION

Fair allocation of collective resources or burdens — jointly referred to as items — between a set of agents is a fundamental task in many multi-agent systems. Its 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).

A growing body of research in the past decade has studied this problem under a canonical model, in which each agent i is assumed to have an *additive* utility function over the set $\mathcal{M}$ of items, denoted $v_i : 2^{\mathcal{M}} \to \mathbb{R}$, satisfying $v_i(S) = \sum_{r \in S} v_i(\{r\})$ for all $S \subseteq \mathcal{M}$. The items being allocated are called *goods* if all agents have non-negative utilities for them, and *chores* if all agents have non-positive utilities for them. The goal is to find a *fair* and *efficient* allocation $\mathbf{x}$, which partitions the set $\mathcal{M}$ of items between the agents, with $\mathbf{x}_i$ denoting the bundle allocated to agent i. Perhaps the most compelling fairness guarantee sought in the literature is *envy-freeness* (EF) [14, 15], 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). This is often sought in conjunction with a basic notion of economic efficiency, called *Pareto optimality* (PO), which demands that no alternative allocation be able to make an agent better off without making another agent worse off (i.e., $\nexists \mathbf{y}$ such that $v_i(\mathbf{y}_i) \geqslant v_i(\mathbf{x}_i)$ for each agent i and at least one inequality is strict). However, it is easy to see that envy-freeness cannot be guaranteed; think of allocating a single item between two agents. Hence, the literature has turned to its relaxations.

An appealing relaxation is called *envy-freeness up to one item* (EF1), which demands that it be possible to remove envy between any two agents by removing a single item from the bundle of one of them. For allocating goods, Caragiannis et al. [12] 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, this rule 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 [28]. Unfortunately, for chores MNW has no natural equivalent, and whether an EF1 and PO allocation of chores always exists has remained a major open question.

To make progress in resolving this open question, we look towards restricted families of utility functions. Perhaps the simplest is the class of *binary* utilities, in which all utilities are in $\{0, -1\}$. For

Authors' addresses: Soroush Ebadian, soroush@cs.toronto.edu, University of Toronto; Dominik Peters, dominik@cs.toronto.edu, University of Toronto; Nisarg Shah, nisarg@cs.toronto.edu, University of Toronto.

allocating goods, the corresponding class of $\{0, 1\}$-utilities has received significant attention [7, 21]. But for allocating chores, this class is trivial: allocating any chore for which some agent has utility 0 to such an agent and dividing the remaining chores (for which all agents have utility -1) as equally as possible yields EF1+PO.

Slightly more complex is the class of *bivalued* utilities, where all utilities 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, 18], where it has been used to achieve stronger fairness guarantees [2]. This class is also of practical interest: 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. Then, one can fix reasonable values of a and b, and assume that all agents have utility a for easy chores and b for difficult ones. For the axiomatic guarantees we are interested in, 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 $p > 1$ (or $\{1, p\}$ for goods). Our main contribution is to show that EF1 and PO allocations of chores always exist under bivalued utilities.

Eliciting bivalued utilities evokes approval voting. A more expressive elicitation method is to ask agents to rank items, potentially with ties. We can interpret a submitted weak order as *weakly lexicographic* utilities. That is, the agent likes each good (resp., hates 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, 22]. We also derive novel existential and computational fairness results for this setting.

1.1 Our Results

Our main result concerns allocation of chores under bivalued utilities, i.e., when, for a common $p > 1$, the utilities of each agent lie in $\{-1, -p\}$. In this case, we prove that an allocation of chores that is both envy-free up to one good (EF1) and Pareto optimal (PO) always exists and can be computed in polynomial time. In fact, we achieve a slightly stronger efficiency guarantee, known as fractional Pareto optimality (fPO) [6], under which not even *fractional* allocations of chores[1] can make an agent better off without making another agent worse off. We obtain this result via an algorithm based on Fisher markets. Our algorithm borrows some ideas from the existing Fisher-market-based algorithm for finding an EF1 and PO allocation of goods [6, 18], but also introduces entirely new techniques that are key to making the algorithm work for chores. Our detailed proof focuses on analyzing and establishing the effectiveness of these new ideas. To the best of our knowledge, ours is the first work to utilize this framework for deriving an existential (and computational) result for allocating chores.

In addition to envy-freeness up to one good (EF1), we consider another popular relaxation of envy-freeness called *maximin share fairness* (MMS) [11]. Here, one aims to provide each agent at least as much utility as the maximum that the agent can achieve by partitioning the items into n bundles and receiving her least preferred bundle from the partition. Here, we consider allocation of goods as well as of chores, under a slightly broader class of utilities, where the utility values of each agent i for the chores (resp., goods) lie in $\{-1, -p_i\}$ (resp., $\{1, p_i\}$) for some $p_i > 1$; note that all agents need not have the same utility values. In this case, we show that an allocation satisfying both MMS and PO always exists and can be computed in polynomial time, if p_i is an integer for each i. Our result, for allocation of both goods and chores, also extends to another class of utilities, namely weakly lexicographic utilities.

[1]Fractional allocations can divide a single chore between multiple agents.

1.2 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. Even when the items are *divisible* (i.e., if they can be portioned out between the agents), these remain well-defined and yield (exact) EF+PO [29]; for goods, they coincide with the maximum Nash welfare rule and can be computed in strongly polynomial time [13, 27], while for chores, they have a more intricate structure [8] and their computation is an open question [10]. In case of indivisible items, Barman et al. [6] use this framework to achieve EF1+PO in pseudo-polynomial time and Garg and Murhekar [18] improve the running time to polynomial when each agent has at most polynomially many utility levels across all bundles of goods. Our work is the first to adapt this framework to allocate indivisible chores instead of goods.

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 being an *integer*, which we refer to as factored bivalued utilities, has recently received attention in the literature on allocating goods in a completely different context. The maximum Nash welfare (MNW) rule is NP-hard to compute for general additive utilities [12], while Barman et al. [7] show that it can be computed in polynomial time for binary ($\{0, 1\}$) utilities. For the more general class of 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 when a and b are coprime.

MMS. For allocating goods, Kurokawa et al. [25] 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 [9], (strictly) lexicographic utilities [22], and ternary ($\{0, 1, 2\}$) utilities [2]. Our MMS result for weakly lexicographic utilities generalizes that of Hosseini et al. [22] and extends it to allocating chores. Also, note that factored bivalued utilities include $\{1, 2\}$-utilities as a special case, and, since we argued in the introduction that 0 utilities can be easily addressed for chores, our MMS result in this case mirrors that 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 [19] and 9/11 for chores [23].

2 PRELIMINARIES

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

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 $\mathbf{v} = (v_1, \dots, v_n)$ 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 $v(r) = v(r')$, 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.

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 $S, S' \subseteq \mathcal{M}$.*

Proof. Let $(L_1, \ldots, L_k)$ be the partition of $\mathcal{M}$ under v as in Definition 2.2. Let $S, S' \subseteq \mathcal{M}$ be two arbitrary subsets of items that $v(S) \leqslant v(S')$. Suppose v is a valuation function for goods.

If $v(S) = v(S')$, then for all $i \in [k]$, $|S \cap L_i| = |S' \cap L_i|$. Therefore, $v'(S) = \sum_{i \in [k]} |S \cap L_i| \cdot m^i = \sum_{i \in [k]} |S' \cap L_i| \cdot m^i = v'(S')$.

If $v(S) < v(S')$, then there exists an $i \in [k]$, such that $|S \cap L_i| < |S' \cap L_i|$, and for all $i' > i$, $|S \cap L_{i'}| = |S' \cap L_{i'}|$. Then,

$$v'(S') - v'(S) = \sum_{j \in [i]} \left(|S' \cap L_j| - |S \cap L_j| \right) \cdot m^j \geqslant m^i - \sum_{j \in [i-1]} |S \cap L_j| \cdot m^j \geqslant m^i - (m-1) \cdot m^{i-1} > 0.$$

The proof for the chores case is similar. □

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. We say that 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). We say that the instance has *personalized bivalued* utilities if each agent i has a bivalued utility function (perhaps with personalized a_i, b_i).[2]

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 [12, 26] and maximin share fairness [11, 25]. These are respectively relaxations of the classical notions of envy-freeness and of proportionality.

[2]Personalized bivalued utilities are also known as 2-ary utilities.

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

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

Note that this is the maximum utility she can obtain by partitioning the items into k bundles and receiving the least valued bundle. We refer to an optimal partition $(S_1, \dots, S_k)$ 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 $v_i(\mathbf{x}_i) \geqslant \mathrm{MMS}_i$ 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. [6] 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 $\mathrm{PB}_i(c) = \frac{|v_i(c)|}{\mathbf{p}(c)}$, and the *minimum pain per buck* (MPB) ratio of agent i is defined as $\mathrm{MPB}_i = \min_{c \in \mathcal{M}} \mathrm{PB}_i(c)$. A chore c with $\mathrm{PB}_i(c) = \mathrm{MPB}_i$ 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*[3] if each agent is allocated only her MPB chores, i.e., if $\mathrm{PB}_i(c) = \mathrm{MPB}_i$ 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.

[3]This is sometimes called a *quasi-equilibrium*, because we do not specify an exogenous budget for each agent.

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

Proof. Let $(\mathbf{x}, \mathbf{p})$ be an equilibrium. Suppose $c \in \mathbf{x}_i$. Then $\mathrm{MPB}_i = \mathrm{PB}_i(c)$ and hence, remembering that $v_i(c)$ is negative, we have $v_i(c)/\mathrm{MPB}_i = v_i(c)/\mathrm{PB}_i(c) = -\mathbf{p}(c)$. On the other hand, if $c \notin \mathbf{x}_i$, then $\mathrm{MPB}_i \leqslant \mathrm{PB}_i(c)$ and hence $v_i(c)/\mathrm{MPB}_i \leqslant v_i(c)/\mathrm{PB}_i(c) = -\mathbf{p}(c)$.

From this it follows that if $c \in \mathbf{x}_i$, then we have $v_i(c)/\mathrm{MPB}_i \geqslant v_j(c)/\mathrm{MPB}_j$ for all $j \in \mathcal{N}$. Hence $\mathbf{x}$ maximizes the value $\sum_{i \in \mathcal{N}} v_i(\mathbf{x}_i)/\mathrm{MPB}_i$. But any Pareto improvement over $\mathbf{x}$ would strictly increase this value (noting that $\mathrm{MPB}_i > 0$), so $\mathbf{x}$ must be 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 [6].

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. [6].

Definition 3.3 (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 [6], pEF1 implies EF1.

Lemma 3.4. *If $(\mathbf{x}, \mathbf{p})$ is a pEF1 equilibrium, then $\mathbf{x}$ is EF1.*

Proof. Fix a pair of agents $i, j \in \mathcal{N}$. We want to show that $v_i(\mathbf{x}_i) \geqslant v_i(\mathbf{x}_j)$. If $\mathbf{x}_i = \emptyset$, this holds trivially. Otherwise, pEF1 indicates that there exists a chore $c \in \mathbf{x}_i$ such that $\mathbf{p}(\mathbf{x}_i \setminus \{c\}) \leqslant \mathbf{p}(\mathbf{x}_j)$. Then, using the definition of PB_i and MPB_i, we have

$$|v_i(\mathbf{x}_i \setminus \{c\})| = \mathrm{MPB}_i \cdot \mathbf{p}(\mathbf{x}_i \setminus \{c\}) \leqslant \mathrm{MPB}_i \cdot \mathbf{p}(\mathbf{x}_j) \leqslant \sum_{c' \in \mathbf{x}_j} \mathrm{PB}_i(c') \cdot \mathbf{p}(c') = |v_i(\mathbf{x}_j)|,$$

where the first transition uses the fact that in an equilibrium allocation agent i is only assigned her MPB chores. Hence, we have $v_i(\mathbf{x}_i \setminus \{c\} \geqslant v_i(\mathbf{x}_j)$, as needed. □

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 $\mathrm{ls} \in \mathcal{N}$ for the *least spender*, i.e., an agent $\mathrm{ls} \in \arg\min_{i \in \mathcal{N}} \mathbf{p}(\mathbf{x}_i)$. Then we see that $(\mathbf{x}, \mathbf{p})$ is pEF1 if and only if $\mathbf{p}_{\text{up to 1}}(\mathbf{x}_i) \leqslant \mathbf{p}(\mathbf{x}_{\mathrm{ls}})$ for all $i \in \mathcal{N}$. Let us call an agent $i \in \mathcal{N}$ a *violator* if $\mathbf{p}_{\text{up to 1}}(\mathbf{x}_i) > \mathbf{p}(\mathbf{x}_{\mathrm{ls}})$. 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 3.5 (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 3.6. *Given a chore division problem $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$ with bivalued utilities, Algorithm 1 finds a PO and EF1 allocation in* $\mathrm{poly}(n, m)$ *time.*

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 $\ell \in (k-2, k-3, \dots, 2, 1)$ 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 $\xleftarrow{c_1} i_1 \xleftarrow{c_2} \cdots \xleftarrow{c_\ell} i_\ell$ 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     $H_k \leftarrow \{i \in \mathcal{N} : \text{there is an agent ls} \in \arg\min_{i\in\mathcal{N}} \mathbf{p}(\mathbf{x}_i) \text{ with ls} \leftsquigarrow i\}$
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)
```

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. [6] 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.

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. Specifically, we prove that for all $k \geqslant 1$ such that Algorithm 1 reaches time $t_{k,a}$,

(H1) $H_k \cap H_\ell = \emptyset$ for all $\ell = 1, \dots, k-1$.
(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 3.7. *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.*

Proof. After initialization in line 3, $(\mathbf{x}, \mathbf{p})$ is an equilibrium because if $c \in \mathbf{x}_i$ then $v_i(c) = \max_{j \in \mathcal{N}} v_j(c)$ since $\mathbf{x}$ is welfare-maximizing. Hence $\text{PB}_i(c) = |v_i(c)|/(p \cdot |v_i(c)|) = 1/p$. On the other hand if $d \notin \mathbf{x}_i$ then $v_i(d) \leqslant \max_{j \in \mathcal{N}} v_j(d)$ so $\text{PB}_i(d) \geqslant 1/p$ by the same calculation. Hence $\text{MPB}_i = 1/p$ and c is an MPB chore for i. So, $(\mathbf{x}, \mathbf{p})$ is an equilibrium.

Item transfers in line 11 of Phase 2a keep $(\mathbf{x}, \mathbf{p})$ in equilibrium because c is an MPB chore for j by (H2). Item transfers in line 19 of Phase 2b preserve equilibrium because c_ℓ is an MPB chore for $i_{\ell-1}$ by the definition of MPB alternating path.

Finally, price changes in line 32 of Phase 3 preserve equilibrium by the definition of α. To see this, note that $\alpha \geqslant 1$ (because, as we have seen, when we set α we are currently in equilibrium, so always $\text{PB}_i(c) \geqslant \text{MPB}_i(c)$ and so $\text{PB}_i(c)/\text{MPB}_i(c) \geqslant 1$). Thus the price change reduces prices, and thus increases some pain-per-buck ratios. It follows that for all $i \in \mathcal{N} \setminus H_k$, items owned by i remain MPB items for i (since MPB_i can only go up and the prices of chores owned by i do not change). Now write MPB'_i and $\text{PB}_i(c)'$ for values after the price reduction. Let $i \in H_k$. We need to prove that all items in $\mathbf{x}_i$ are MPB items for i after the price change. First we claim that $\text{MPB}'_i = \alpha\, \text{MPB}_i$. For $c \in \mathbf{x}(\mathcal{N} \setminus H_k)$, we have by choice of α that

$$\text{PB}'_i(c) = \text{PB}_i(c) = \frac{\text{PB}_i(c)}{\text{MPB}_i} \text{MPB}_i \geqslant \alpha\, \text{MPB}_i\,.$$

For all $c \in \mathbf{x}(H_k)$,

$$\text{PB}'_i(c) = \alpha\, \text{PB}_i(c) \geqslant \alpha\, \text{MPB}_i\,.$$

Finally for all $c \in \mathbf{x}_i$ we have $\mathrm{PB}_i(c) = \mathrm{MPB}_i$ since c was an MPB item for i before the price change. Hence

$$\mathrm{PB}'_i(c) = \alpha\,\mathrm{PB}_i(c) = \alpha\,\mathrm{MPB}_i\,.$$

From these, it follows that indeed $\mathrm{MPB}'_i = \alpha\,\mathrm{MPB}_i$, and that $\mathrm{PB}'_i(c) = \mathrm{MPB}'_i$ for all $c \in \mathbf{x}_i$. So all items owned by i are MPB items for i after the price change, as required. □

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. [6]. The proof of this result is deferred to the appendix.

Lemma 3.8 (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 3.9. *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.*

Proof. Every step of the algorithm is well-defined. This is obvious except for line 11, where the algorithm implicitly asserts the existence of a chore satisfying a certain property. But by (H2) such a chore exists every time line 11 is reached.

The only way that the algorithm can terminate is if $(\mathbf{x}, \mathbf{p})$ is pEF1 (line 24), at which time it is also an equilibrium by Lemma 3.7. So, it suffices to show that the algorithm terminate in polynomial time.

Consider an execution of Phase 2a. For any value of ℓ, consider the while loop in line 16. In each step of the while loop, a chore is transferred from an agent in H_ℓ to an agent in H_t for some $t > \ell$. Since chores only move from lower-numbered H-sets to higher-numbered H-sets, each item can be moved at most once. Hence, this while loop terminates after at most m steps, and hence, Phase 2a terminates in polynomial time.

Phase 2b terminates in polynomial time by Lemma 3.8(a) which we can apply since $(\mathbf{x}, \mathbf{p})$ is an equilibrium by Lemma 3.7.

Phase 3 can be executed at most n times, because in each execution at least one agent (the least spender ls) is assigned to a set H_k, and that agent was not previously assigned to such a set by (H1). Since there are only n agents, this can happen at most n times. □

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

Proof. (H1) holds vacuously. (H2) also holds vacuously because line 11 is never reached in iteration 1, because the for-loop of Phase 2a is not executed. (H3) holds because otherwise Phase 2b would not have stopped. (H4) holds by the definition of entitled(i).

Call a chore *very difficult* if $v_i(c) = -p$ for all $i \in \mathcal{N}$. In line 3, we set prices to be p^2 for very difficult chores, and p for other chores.

Consider time $t_{1,b}$, when prices are the same as at initialization. Note that for all $i \in \mathcal{N}$, we have $\mathrm{MPB}_i = 1/p$, because we have assumed that every i values at least one item c at -1, so c is not very difficult and $\mathbf{p}(c) = p$ giving $\mathrm{PB}_i(c) = 1/p$. (Clearly MPB_i cannot be less than $1/p$ since the only possible pain-per-buck ratios are p/p, $1/p$, and p/p^2. The ratio $1/p^2$ is not possible since only very difficult chores have price p^2.) Let c be a very difficult item. Then $\mathrm{PB}_i(c) = p/p^2 = 1/p$ for all $i \in \mathcal{N}$. Hence c is an MPB chore for all agents. It follows that if i is the owner of c at time $t_{1,b}$, then $i \in H_1$. Thus, at $t_{1,b}$ all very difficult chores are owned by agents in H_1. Next, let $c \in \cup_{i' \in \mathcal{N} \setminus H_1} \mathbf{x}_{i'}$ be a chore not owned by an agent in H_1, say $c \in \mathbf{x}_j$. Then for all $i \in H_1$ we must have $\mathrm{PB}_i(c) > \mathrm{MPB}_i$ or else c is an MPB chore for i and then we would have $j \in H_1$ by the definition of H_1. Hence $\mathrm{PB}_i(c) = 1$. It follows that in line 28, we set

$$\alpha = \min \{\mathrm{PB}_i(c)/\mathrm{MPB}_i : i \in H_1, c \in \mathbf{x}(\mathcal{N} \setminus H_1)\} = \tfrac{1}{1/p} = p.$$

This gives (H5).

Next, in line 32, we multiply the price of each item owned by H_1 by $1/\alpha$. In particular, we update the price of every very difficult item from p^2 to p, and we may update some other chores' prices from p to 1. After this update at time $t_{1,a}$, we thus have $\mathbf{p}(c) \in \{1, p\}$ for all chores $c \in \mathcal{M}$. Also, any item c that is now priced 1 must have had its price updated, so c is owned by someone in H_1, and hence $c \in \mathrm{entitled}(i)$ for some $i \in H_1$. This gives (H6).

Finally, we calculate the values of MPB_i after the price change, i.e. at time $t_{1,a}$. Let $i \in H_1$. There exists some item c with $v_i(c) = -1$. Before the price change, c was an MPB chore for i. Thus by the definition of H_1, the owner of c is in H_1. Now c's price has changed from p to 1. Thus if $v_i(c) = -1$ then $\mathbf{p}(c) = 1$, and $\mathrm{PB}_i(c) = 1$. On the other hand, for chores c with $v_i(c) = -p$, we have $\mathbf{p}(c) \leqslant p$ and so $\mathrm{PB}_i(c) \geqslant 1$. It follows that $\mathrm{MPB}_i = 1$ after the price change. Next let $j \in \mathcal{N} \setminus H_1$. Note that j was not a least spender at $t_{1,b}$ (because least spenders are in H_1). Hence $\mathbf{p}(\mathbf{x}_j) > 0$ and so $\mathbf{x}_j \neq \emptyset$. Take some $c \in \mathbf{x}_j$. At $t_{1,b}$, we had $\mathrm{MPB}_j = 1/p$, so $\mathrm{PB}_j(c) = 1/p$. The price of c did not change, because c is not owned by H_1. So also at $t_{1,a}$, we have $\mathrm{PB}_j(c) = 1/p$ and hence $\mathrm{MPB}_j = 1/p$ because $1/p$ is the smallest possible pain-per-buck ratio. This gives (H7). □

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 shows that (H2) holds for iteration $k+1$.

Lemma 3.11. *During iteration $k+1$, each time the algorithm reaches line 11, there exists a chore $c \in \mathbf{x}_i \setminus \mathrm{entitled}(i)$, and any such chore is an MPB chore for the agent j identified in line 9.*

Before we prove Lemma 3.11, we need two additional results.

Lemma 3.12. *For all $1 \leqslant \ell < k$, we have $\mathbf{x}^{t_{\ell,b}}(H_{\ell+1} \cup \ldots \cup H_k) \subseteq \mathbf{x}^{t_{k,b}}(H_{\ell+1} \cup \ldots \cup H_k)$.*

Proof. Consider some agent $s \in H_r \subseteq H_{\ell+1} \cup \ldots \cup H_k$ and some item $c \in \mathbf{x}_s^{t_{\ell,b}}$. Suppose that the price of item c changes at some iteration u with $\ell + 1 \leqslant u \leqslant k$. Then $c \in \mathrm{entitled}(i)$ for some $i \in H_u$, and then by (H4) we have $c \in \mathbf{x}_i^{t_{k,b}}$ as desired. Otherwise, the price of c does not change. Consider the owner of c at time $t_{r,b}$, say $c \in \mathbf{x}_j^{t_{r,b}}$. Then $j \notin H_r$ since the price of c did not change. Now, at time $t_{\ell,b}$, agent s owned c, so at that time $\mathrm{PB}_s(c) = \mathrm{MPB}_s$ (because the algorithm is always in equilibrium by Lemma 3.7). Since the price of c has not changed, and since the value of MPB_s has continued to be $1/p$ by (H7) applied to iteration $r-1$, we still have $\mathrm{PB}_s(c) = \mathrm{MPB}_s$ at time $t_{r,b}$. But then since $s \in H_r$, we also have $j \in H_r$, a contradiction. □

The next lemma is a sort of load balancing lemma. Intuitively, it says that if a group of agents are allocated some chores of equal price, and over the time, they receive more chores and the prices of the chores increase, then regardless of how the distribution of those chores between the agents changes, the minimum spending in the group can only increase.

Lemma 3.13. *Let $\mathcal{N}$ be a set of agents. Let $\mathcal{M}$ and $\mathcal{M}'$ be two sets of chores with $|\mathcal{M}| \leqslant |\mathcal{M}'|$. Let $(\mathbf{x}, \mathbf{p})$ and $(\mathbf{x}', \mathbf{p}')$ be pEF1 equilibria, where $\mathbf{x}$ and $\mathbf{x}'$ are allocations of $\mathcal{M}$ and $\mathcal{M}'$, respectively, to the agents in $\mathcal{N}$. Suppose that $\mathbf{p}(c) = 1$ for all $c \in \mathcal{M}$ and that $\mathbf{p}'(c') \in \{1, p\}$ for all $c' \in \mathcal{M}'$. Then $\min_{i \in \mathcal{N}} \mathbf{p}(\mathbf{x}_i) \leqslant \min_{i \in \mathcal{N}} \mathbf{p}'(\mathbf{x}'_i)$.*

Proof. Let the least spenders of $\mathbf{x}$ and $\mathbf{x}'$ be ls_1 and ls_2 respectively. Let $k = \lfloor |\mathcal{M}|/n \rfloor$. Because $\mathbf{x}$ is pEF1 and all items are priced 1, then each agent is allocated k or $k+1$ items in $\mathbf{x}$ and so $\mathbf{p}(\mathbf{x}_{\mathrm{ls}_1}) = k$.

For a contradiction, assume that $\mathbf{p}'(\mathbf{x}'_{\mathrm{ls}_2}) < \mathbf{p}(\mathbf{x}_{\mathrm{ls}_1}) = k$. Then $|\mathbf{x}'_{\mathrm{ls}_2}| < k$. For other agents $o \in \mathcal{N} \setminus \{\mathrm{ls}_2\}$, we have $|\mathbf{x}'_o| - 1 \leqslant \mathbf{p}'_{\text{up to } 1}(\mathbf{x}'_o) \leqslant \mathbf{p}'(\mathbf{x}'_{\mathrm{ls}_2}) < k$, where the first inequality holds because $\mathbf{p}'(c) \geqslant 1$ for all c, and the second holds because $\mathbf{x}'$ is pEF1. Thus $|\mathbf{x}'_o| - 1 < k$ and so $|\mathbf{x}'_o| \leqslant k$. Now,

$$|\mathcal{M}'| = |\mathbf{x}'_{\mathrm{ls}_2}| + \sum_{o \in \mathcal{N} \setminus \{ls_2\}} |\mathbf{x}'_o| \leqslant k - 1 + (n-1)k = nk - 1,$$

However, $|\mathcal{M}'| \geqslant |\mathcal{M}| \geqslant nk$, which is a contradiction. □

We are now ready to prove Lemma 3.11.

Proof of Lemma 3.11. We prove the second part first. Any $c \in \mathbf{x}_i \setminus \mathrm{entitled}(i)$ must have $\mathbf{p}(c) = p$ by (H6) for iteration k. Since $j \in H_1 \cup \cdots \cup H_k$, we have $\mathrm{MPB}_j = 1$ by (H7) applied to iteration k. Thus $\mathrm{PB}_j(c)$ cannot be $1/p$, so $\mathrm{PB}_j(c) = 1 = \mathrm{MPB}_j$. Hence c is an MPB chore for j.

For the first part, assume that there is some time t when the algorithm select agents $i \in H_\ell$ and $j \in H_{\ell+1} \cup \cdots \cup H_k$ where no $c \in \mathbf{x}_i^t \setminus \mathrm{entitled}(i)$ exists. Let t be the first such time. By line 10, we have $\mathbf{p}^t_{\text{up to } 1}(\mathbf{x}^t_i) > \mathbf{p}^t(\mathbf{x}^t_j)$. By (H4), we had $\mathrm{entitled}(i) \subseteq \mathbf{x}_i^{t_{k,b}}$. Since t is the time of first failure, so far no entitled item has been transferred away from i. Thus $\mathbf{x}^t_i = \mathrm{entitled}(i)$. By definition of $\mathrm{entitled}(i)$ and since $i \in H_\ell$, thus $\mathbf{x}^t_i = \mathbf{x}_i^{t_{\ell,a}}$. Now we have

$$\begin{aligned}
\mathbf{p}^t_{\text{up to } 1}(\mathbf{x}^t_i) &= \mathbf{p}^{t_{\ell,a}}_{\text{up to } 1}(\mathbf{x}_i^{t_{\ell,a}}) && \text{(since prices of entitled chores have not changed after } t_{\ell,a} \text{ by (H1))}\\
&= \tfrac{1}{p} \cdot \mathbf{p}^{t_{\ell,b}}_{\text{up to } 1}(\mathbf{x}_i^{t_{\ell,b}}) && \text{(since } \alpha = \tfrac{1}{p} \text{ in iteration } \ell \text{ by (H5))}\\
&\leqslant \tfrac{1}{p} \cdot \min\nolimits_{o \in \mathcal{N}} \mathbf{p}^{t_{\ell,b}}(\mathbf{x}_o^{t_{\ell,b}}). && \text{(since } i \in H_\ell \text{ was not a violator at } t_{\ell,b} \text{ by (H3))}
\end{aligned}$$

By Lemma 3.12, $\mathbf{x}^{t_{\ell,b}}(H_{\ell+1} \cup \ldots \cup H_k) \subseteq \mathbf{x}^{t_{k,b}}(H_{\ell+1} \cup \ldots \cup H_k)$. And, so far in Phase 2a of iteration $k+1$, no item has been transferred out of $H_{\ell+1} \cup \ldots \cup H_k$. Therefore, $\mathbf{x}^{t_{k,b}}(H_{\ell+1} \cup \ldots \cup H_k) \subseteq \mathbf{x}^t(H_{\ell+1} \cup \ldots \cup H_k)$. By (H6), at $t_{\ell,b}$, all chores owned by $H_{\ell+1} \cup \ldots \cup H_k$ were priced p. Now, at t, they own a superset of those chores with prices of 1 or p. By applying Lemma 3.13 with the first set of chores being $\mathbf{x}^{t_{\ell,b}}(H_{\ell+1} \cup \ldots \cup H_k)$ all priced 1, and the second set being $\mathbf{x}^{t'}(H_{\ell+1} \cup \ldots \cup H_k)$ with their current prices at time t', we can conclude that

$$\frac{1}{p} \cdot \min_{o \in H_{\ell+1} \cup \ldots \cup H_k} \mathbf{p}^{t_{\ell,b}}(\mathbf{x}_o^{t_{\ell,b}}) \leqslant \min_{o \in H_{\ell+1} \cup \ldots \cup H_k} \mathbf{p}^t(\mathbf{x}^t_o) = \mathbf{p}^t(\mathbf{x}^t_j),$$

where the last equality holds by choice of j. Combining this with the previous inequalities, we get $\mathbf{p}^t_{\text{up to } 1}(\mathbf{x}^t_i) \leqslant \mathbf{p}^t(\mathbf{x}^t_j)$, a contradiction. □

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 3.14. *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.*

Proof. For an allocation $\mathbf{x}$ and sets $S, T \subseteq \mathcal{N}$, let us write

$$\mathrm{NoViol}(\mathbf{x}, S \to T) \iff \mathbf{p}_{\text{up to } 1}(\mathbf{x}_i) \leqslant \mathbf{p}(\mathbf{x}_j) \text{ for all } i \in S \text{ and } j \in T.$$

In this notation, an agent i is not a violator if and only if $\mathrm{NoViol}(\mathbf{x}, \{i\} \to \mathcal{N})$.

For each $\ell \in [k]$, write $R_\ell = H_\ell \cup \cdots \cup H_k$. By (H3) applied to iteration k, no agent in R_ℓ is a violator at time $t_{k,b}$. In particular, this means that at time $t_{k,b}$ we have $\mathrm{NoViol}(\mathbf{x}, R_\ell \to \mathcal{N} \setminus R_\ell)$ for each $\ell \in [k]$. The same is true at time $t_{k,a}$, because the price reduction reduces the prices of goods held by R_ℓ but not those held by $\mathcal{N} \setminus R_\ell$. So we have

$$\mathrm{NoViol}(\mathbf{x}, R_\ell \to \mathcal{N} \setminus R_\ell) \quad \text{for all } \ell \in [k], \text{ at time } t_{k,a}. \tag{1}$$

Induction on ℓ. Now, we prove inductively for $\ell = k-1, \dots, 1$, that at the end of the for-loop iteration corresponding to ℓ, we have

$$\mathrm{NoViol}(\mathbf{x}^\ell, R_\ell \to \mathcal{N}), \tag{2}$$

where $\mathbf{x}^\ell$ is the allocation at the end of the ℓ-th iteration of the for-loop of Phase 2a.

As a base case, we take $\ell = k$, noting that just before the for-loop starts (i.e. at time $t_{k,a}$), we have $\mathrm{NoViol}(\mathbf{x}^k, R_k \to \mathcal{N})$ because

- from (1) we know that $\mathrm{NoViol}(\mathbf{x}^k, R_k \to \mathcal{N} \setminus R_k)$, and
- from (H3) we had $\mathrm{NoViol}(\mathbf{x}^k, H_k \to H_k)$ at time $t_{k,b}$, and since the price reduction changes the prices of all chores held by H_k by the same factor α, we also have $\mathrm{NoViol}(\mathbf{x}^k, H_k \to H_k)$ at time $t_{k,a}$. Since $R_k = H_k$, hence $\mathrm{NoViol}(\mathbf{x}^k, R_k \to R_k)$.

Once we have established the induction step, we can apply (2) for $\ell = 1$ to get that $R_1 = H_1 \cup \cdots \cup H_k$ are not violators at t_{mid}, which is the lemma statement.

Suppose that (2) holds for ℓ at the start of iteration $\ell - 1$. (This is true either by the base case, or because (2) for ℓ held at the end of iteration ℓ and hence also at the start of iteration $\ell - 1$.) Our goal is to show that (2) holds for $\ell - 1$ at the end of iteration $\ell - 1$. We prove this in two steps: first, we show $\mathrm{NoViol}(\mathbf{x}^{\ell-1}, R_{\ell-1} \to \mathcal{N} \setminus R_{\ell-1})$, and then we show $\mathrm{NoViol}(\mathbf{x}^{\ell-1}, R_{\ell-1} \to R_{\ell-1})$.

No violators to $\mathcal{N} \setminus R_{\ell-1}$. For all agents $s \in \mathcal{N} \setminus R_{\ell-1}$, at the start of iteration $\ell - 1$ we have

- $\max_{i \in R_\ell} \mathbf{p}_{\text{up to }1}(\mathbf{x}_i^\ell) \leqslant \mathbf{p}(\mathbf{x}_s^\ell)$ by the induction hypothesis (2), and
- $\max_{i \in H_{\ell-1}} \mathbf{p}_{\text{up to }1}(\mathbf{x}_i^\ell) \leqslant \mathbf{p}(\mathbf{x}_s^\ell)$ by (1), noting that the bundles of $H_{\ell-1}$ and of $\mathcal{N} \setminus R_{\ell-1}$ have not been changed since $t_{k,a}$.

Hence, $\max_{i \in R_{\ell-1}} \mathbf{p}_{\text{up to }1}(\mathbf{x}_i^\ell) \leqslant \mathbf{p}(\mathbf{x}_s^\ell)$.

During the execution of iteration $\ell-1$ of Phase 2a, the value $\max_{i \in R_{\ell-1}} \mathbf{p}_{\text{up to }1}(\mathbf{x}_i)$ never increases: This is because if we transfer c from agent i to agent j, then

- $\mathbf{p}_{\text{up to }1}(\mathbf{x}_i)$ decreases because i gave away an item,
- $\mathbf{p}_{\text{up to }1}(\mathbf{x}_j)$ increases but not too much: Write $\mathbf{x}$ and $\mathbf{x}'$ for the allocations before and after the transfer, and recall that $\mathbf{p}_{\text{up to }1}(\mathbf{x}_i) > \mathbf{p}(\mathbf{x}_j)$ since we performed the transfer. Then

$$\mathbf{p}_{\text{up to }1}(\mathbf{x}'_j) = \mathbf{p}_{\text{up to }1}(\mathbf{x}_j \cup \{c\}) \leqslant \mathbf{p}(\mathbf{x}_j) < \mathbf{p}_{\text{up to }1}(\mathbf{x}_i).$$

  Hence $\mathbf{p}_{\text{up to }1}(\mathbf{x}'_j)$ is smaller than the previous maximum value of $\mathbf{p}_{\text{up to }1}$.

Recall that $\mathbf{x}^{\ell-1}$ is the allocation at the end of iteration $\ell - 1$. Thus for all $s \in \mathcal{N} \setminus R_{\ell-1}$,

$$\max_{i \in R_{\ell-1}} \mathbf{p}_{\text{up to }1}(\mathbf{x}_i^{\ell-1}) \leqslant \max_{i \in R_{\ell-1}} \mathbf{p}_{\text{up to }1}(\mathbf{x}_i^\ell) \leqslant \mathbf{p}(\mathbf{x}_s^\ell) = \mathbf{p}(\mathbf{x}_s^{\ell-1})$$

where the last equality holds because the bundle $\mathbf{x}_s$ has not changed since the start of iteration $\ell - 1$. Therefore, $\mathrm{NoViol}(\mathbf{x}^{\ell-1}, R_{\ell-1} \to \mathcal{N} \setminus R_{\ell-1})$.

No violators for $R_{\ell-1}$. At the start of iteration $\ell-1$ we have all of the following:

(a) $\text{NoViol}(\mathbf{x}^\ell, H_{\ell-1} \to H_{\ell-1})$ by (H3) since the bundles of $H_{\ell-1}$ have not changed since $t_{k,b}$,
(b) $\text{NoViol}(\mathbf{x}^\ell, R_\ell \to R_\ell)$ by inductive hypothesis (2),
(c) $\text{NoViol}(\mathbf{x}^\ell, R_\ell \to H_{\ell-1})$ by inductive hypothesis (2).

We now show inductively that after each transfer, we still have (a), (b), and (c).

So suppose (a), (b), and (c) hold for allocation $\mathbf{x}$. We now transfer item c from $i \in H_{\ell-1}$ to $j \in R_\ell$, obtaining allocation $\mathbf{x}'$. We show that (a), (b), and (c) also hold for allocation $\mathbf{x}'$.

(a) For all $s \in H_{\ell-1}$,
$$\mathbf{p}_{\text{up to }1}(\mathbf{x}'_s) \leqslant \mathbf{p}_{\text{up to }1}(\mathbf{x}_s) \leqslant \mathbf{p}_{\text{up to }1}(\mathbf{x}_i) \leqslant \mathbf{p}(\mathbf{x}_i) - \mathbf{p}(c) = \mathbf{p}(\mathbf{x}'_i),$$
where the first inequality holds because $H_{\ell-1}$ did not receive items, and the second inequality holds by choice of i. Hence, $\text{NoViol}(\mathbf{x}', H_{\ell-1} \to \{i\})$. Because (a) held before the transfer, because the transfer did not change the bundles for $H_{\ell-1} \setminus \{i\}$, and because $\mathbf{p}_{\text{up to }1}(\mathbf{x}'_i) \leqslant \mathbf{p}_{\text{up to }1}(\mathbf{x}_i)$, we have $\text{NoViol}(\mathbf{x}', H_{\ell-1} \to H_{\ell-1} \setminus \{i\})$. Putting the two together, we have $\text{NoViol}(\mathbf{x}', H_{\ell-1} \to H_{\ell-1})$.

(b) For all $s \in R_\ell$,
$$\mathbf{p}_{\text{up to }1}(\mathbf{x}'_j) \leqslant \mathbf{p}(\mathbf{x}_j \cup \{c\}) - \mathbf{p}(c) = \mathbf{p}(\mathbf{x}_j) \leqslant \mathbf{p}(\mathbf{x}_s) \leqslant \mathbf{p}(\mathbf{x}'_s),$$
where the penultimate inequality holds by choice of j and the last because R_ℓ does not give away items. Therefore, $\text{NoViol}(\mathbf{x}', \{j\} \to R_\ell)$. Because (b) held before the transfer, because the allocation did not change for $R_\ell \setminus \{j\}$, and because $\mathbf{p}(\mathbf{x}'_j) \geqslant \mathbf{p}(\mathbf{x}_j)$, we have $\text{NoViol}(\mathbf{x}', R_\ell \setminus \{j\} \to R_\ell)$. Putting the two together, we have $\text{NoViol}(\mathbf{x}', R_\ell \to R_\ell)$.

(c) Note first that since c is not an entitled chore, we have $\mathbf{p}(c) = p$ due to (H6), so c is a chore with maximum price. Therefore
$$\mathbf{p}_{\text{up to }1}(\mathbf{x}'_j) = \mathbf{p}(\mathbf{x}_j \cup \{c\}) - \mathbf{p}(c) = \mathbf{p}(\mathbf{x}_j) \quad \text{and} \quad \mathbf{p}_{\text{up to }1}(\mathbf{x}_i) = \mathbf{p}(\mathbf{x}_i) - \mathbf{p}(c) = \mathbf{p}(\mathbf{x}'_i). \tag{3}$$
We show that (c) holds for $\mathbf{x}'$ in four parts.
- Because (c) held before the transfer, we have $\text{NoViol}(\mathbf{x}, R_\ell \setminus \{j\} \to H_{\ell-1} \setminus \{i\})$. Since the transfer did not change the bundles of $R_\ell \setminus \{j\}$ and of $H_{\ell-1} \setminus \{i\}$, we thus have $\text{NoViol}(\mathbf{x}', R_\ell \setminus \{j\} \to H_{\ell-1} \setminus \{i\})$.
- Using (3), and because we performed the transfer, we have
$$\mathbf{p}_{\text{up to }1}(\mathbf{x}'_j) = \mathbf{p}(\mathbf{x}_j) < \mathbf{p}_{\text{up to }1}(\mathbf{x}_i) = \mathbf{p}(\mathbf{x}'_i). \tag{4}$$
Hence $\text{NoViol}(\mathbf{x}', \{j\} \to \{i\})$.
- For all $s \in H_{\ell-1} \setminus \{i\}$, we have
$$\mathbf{p}_{\text{up to }1}(\mathbf{x}'_j) \overset{(4)}{<} \mathbf{p}(\mathbf{x}'_i) \overset{(3)}{=} \mathbf{p}_{\text{up to }1}(\mathbf{x}_i) \leqslant \mathbf{p}(\mathbf{x}_s) = \mathbf{p}(\mathbf{x}'_s)$$
which shows $\text{NoViol}(\mathbf{x}', \{j\} \to H_{\ell-1} \setminus \{i\})$.
- For all $s \in R_\ell \setminus \{j\}$, we have
$$\mathbf{p}_{\text{up to }1}(\mathbf{x}'_s) = \mathbf{p}_{\text{up to }1}(\mathbf{x}_s) \leqslant \mathbf{p}(\mathbf{x}_j) \overset{(3)}{=} \mathbf{p}_{\text{up to }1}(\mathbf{x}'_j) \overset{(4)}{<} \mathbf{p}(\mathbf{x}'_i)$$
which shows $\text{NoViol}(\mathbf{x}', R_i \setminus \{j\} \to \{i\})$.

Putting all these together, we get $\text{NoViol}(\mathbf{x}', R_\ell \to H_{\ell-1})$, which is (c).

The induction tells us that (a), (b), and (c) hold when iteration $\ell-1$ finishes, i.e. they hold for $\mathbf{x}^{\ell-1}$. In addition, because the iteration ended (line 10), we must have $\text{NoViol}(\mathbf{x}^{\ell-1}, H_{\ell-1} \to R_\ell)$. This together with (a), (b), and (c) gives $\text{NoViol}(\mathbf{x}^{\ell-1}, R_{\ell-1} \to R_{\ell-1})$, as desired. ◻

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

Lemma 3.15. *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.*

Proof. Write $R = H_1 \cup \cdots \cup H_k$. Recall that at the start of Phase 2b, no agent in R was a violator (Lemma 3.14). Thus, for $i \in R$ to give away a chore in Phase 2b, i needs to become a violator. This can only happen if i receives a chore during Phase 2b.

Consider a transfer during Phase 2b of a chore c from $s \notin R$ to $i \in R$ at time t, after which i becomes a violator. At time t, there was an MPB alternating path $\mathrm{ls} \leftsquigarrow i \xleftarrow{c} s$ of length $\ell + 1$, say. (At this time, i cannot be a least spender, because a least spender cannot become a violator after being given one item.) Since Phase 2b chooses MPB alternating paths of minimum length, there was no suitable path of length ℓ available. At time t' (the time step immediately after the transfer), there is an MPB alternating path $\mathrm{ls} \leftsquigarrow i$ of length ℓ, which means that i is the violator uniquely closest to ls at this point. It follows that at t', we perform a transfer of some item c' from i to some agent j, using an MPB alternating path $\mathrm{ls} \leftsquigarrow j \xleftarrow{c'} i$ of length ℓ. We now claim that $j \notin R$. This is because if j was a member of R, then at time t, we would have $j \xleftarrow{c} s$ (because $\mathbf{p}(c) = p$ by (H6) and thus $\mathrm{PB}_j(c) = 1 = \mathrm{MPB}_j$). Thus $\mathrm{ls} \leftsquigarrow j \xleftarrow{c} s$ would have been an MPB alternating path of length ℓ to violator s at time t, contradicting that the shortest such path had length $\ell + 1$. Hence $j \notin R$. Now, because we performed the transfer $j \xleftarrow{c'} i$, item c' is an MPB chore for j. Thus from (H7), $\mathrm{PB}_j(c') = 1/p$. It follows that $\mathbf{p}(c') = p$ and $v_j(c') = -1$. But then $c' \notin \mathrm{entitled}(i)$, because the only entitled chores with price p are very difficult chores that every agent values at $-p$ (see the proof of Lemma 3.10), and $v_j(c') \neq -p$ so c' is not a very difficult item. Thus $c' \notin \mathrm{entitled}(i)$. Recall that t is the time before the transfer of c from s to i, that t' is the time after that transfer, and let t'' be the time after the transfer of c' from i to j. (So $\mathbf{x}_i^{t''} = \mathbf{x}_i^t \cup \{c\} \setminus \{c'\}$.) We now show that agent i is not a violator anymore at time t''. First, note that $\mathbf{x}_i^{t''}$ is obtained from $\mathbf{x}_i^t$ by adding an item priced p and removing an item priced p. Thus $\mathbf{p}_{\text{up to }1}(\mathbf{x}_i^{t''}) = \mathbf{p}_{\text{up to }1}(\mathbf{x}_i^t)$. Since i was not a violator at time t (so $\mathbf{p}_{\text{up to }1}(\mathbf{x}_i^t) \leqslant \min_{s \in \mathcal{N}} \mathbf{p}(\mathbf{x}_s)$), and least spending cannot have decreased since time t by Lemma 3.8(2), it follows that i is not a violator at time t''.

Thus we have shown that if $i \in R$ becomes a violator due to being given an item from some agent $s \notin R$, then i ceases to be a violator in the immediately next step by giving away a non-entitled item to an agent $j \notin R$. The only other way that $i \in R$ could give away an item is if i becomes a violator due to being given an item from some agent $s \in R$, but we show this never happens: if it did, consider the first time some $i \in R$ becomes a violator in this way. But then $s \in R$ must have previously become a violator due to being given an item from some $s' \notin R$. But we have already proven that this is a contradiction, because s' will immediately become a non-violator without giving an item to a member of R. □

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

Lemma 3.16. *Suppose Algorithm 1 reaches time $t_{k+1,a}$. Then (H1) to (H7) hold for $k+1$.*

Proof. Write $R = H_1 \cup \cdots \cup H_k$. We have already proved (H2) in Lemma 3.11. For (H3), by Lemma 3.15 no agent in R is a violator at the end of Phase 2b, and thus at time $t_{k+1,b}$, which is (H3).

For (H1), note that because the algorithm has reached time $t_{k+1,b}$, it did not return an allocation in line 24. Thus there exists a violator s. As we just proved, $s \notin R$. By inductive hypothesis (H6), at time $t_{k,a}$, we had $\mathbf{p}(c) = p$ for all $c \in \mathbf{x}_s^{t_{k,a}}$. Because $s \notin R$, the bundle $\mathbf{x}_s$ is not changed during Phase 2a of iteration $k+1$. By Lemma 3.15, during Phase 2b, no entitled items are ever transferred. Thus using inductive hypothesis (H6), during Phase 2b only items priced p are transferred. So all chores owned by s at $t_{k+1,b}$ are priced p. Note that $\mathbf{x}_s \neq \emptyset$, because we had $\mathbf{x}_s \neq \emptyset$ at $t_{1,a}$ because $i \notin H_1$ (see the proof of Lemma 3.10) and Phases 2a and 2b never take away an agent's last item. So select some chore $c \in \mathbf{x}_s$. We show that $R \cap H_{k+1} = \emptyset$. Let $i \in R$. We wish to prove that $i \notin H_{k+1}$. By inductive hypothesis (H7), we had $\mathrm{MPB}_i = 1$ at time $t_{k,a}$ and hence also at $t_{k+1,b}$. Then $\mathrm{PB}_i(c) = 1$ because $\mathbf{p}(c) = p$ (and $\mathrm{PB}_i(c) = 1/p$ would contradict $\mathrm{MPB}_i = 1$) and thus c is an MPB item for i. If we had $i \in H_{k+1}$, then (due to item c) also $s \in H_{k+1}$ by definition of H_{k+1}. But this is a contradiction because violators cannot be in H_{k+1} since Phase 2b then would not have terminated. Hence $i \notin H_{k+1}$. Thus $R \cap H_{k+1} = \emptyset$, which is (H1).

For (H4), note first that at time $t_{k+1,a}$, for all $i \in H_{k+1}$, the algorithm has just set $\mathrm{entitled}(i) = \mathbf{x}_i$ and thus $\mathrm{entitled}(i) \subseteq \mathbf{x}_i$. For $i \in R$, note that we had $\mathrm{entitled}(i) \subseteq \mathbf{x}_i$ at time $t_{k,a}$ by inductive hypothesis. Phase 2a never transfers an entitled item of i. By Lemma 3.15, Phase 2b does not do this either. So we have proven (H4).

Next, check (H5). We have shown that in iteration $k+1$, no entitled chores were transferred. Thus all chores that were transferred had price p at time $t_{k,a}$ (and thus also at time $t_{k+1,b}$) by inductive hypothesis (H6). We now prove (H5), that $\alpha = p$ in iteration $k+1$. Let $c \in \mathbf{x}^{t_{k+1,b}}(\mathcal{N} \setminus (H_1 \cup \cdots \cup H_{k+1}))$ be a chore not owned by $R \cup H_{k+1}$ at time $t_{k+1,b}$; say it is owned by s. (Such a chore exists because otherwise the algorithm would have terminated at line 24.) We have $\mathbf{p}(c) = p$. Item c cannot be an MPB chore for anyone in H_{k+1} since otherwise $s \in H_{k+1}$, contradiction. Since $\mathrm{MPB}_i = 1/p$ for all $i \in H_{k+1}$, we must have $\mathrm{PB}_i(c) = 1$. Hence we have $\alpha = 1/(1/p) = p$, giving (H5).

For (H6), we only need to consider chores owned by H_{k+1}, since no other chores have their price changed. Let $c \in \mathbf{x}_i^{t_{k+1,b}}$ for some $i \in H_{k+1}$. Since $\mathbf{p}^{t_{k+1,b}}(c) \in \{1, p\}$ by inductive hypothesis (H6) and $\mathrm{MPB}_i = 1/p$ at time $t_{k+1,b}$ by (H7), we have $\mathbf{p}^{t_{k+1,b}}(c) = p$. Since $\alpha = p$ as we have just shown, then $\mathbf{p}^{t_{k+1,a}}(c) = \frac{1}{\alpha}p = 1$. Because $c \in \mathrm{entitled}(i)$, this gives (H6).

Finally for (H7), since chores owned by H_{k+1} changed price from p to 1, for each $i \in H_{k+1}$ the value of MPB_i changes from $1/p$ to 1. For other agents, the MPB values have not changed: For $i \in \mathcal{N} \setminus (H_1 \cup \cdots \cup H_{k+1})$ we had $\mathrm{MPB}_i = 1/p$ at time $t_{k,a}$ by inductive hypothesis (H7). At time $t_{k+1,b}$, agent i owns at least one item c (since $i \notin H_1$) and since the algorithm always stays in equilibrium, $\mathrm{PB}_i(c) = 1/p$. Since the price of c was not changed in iteration $k+1$, we also have $\mathrm{PB}_i(c) = 1/p$ at time $t_{k+1,a}$. Hence at this time, $\mathrm{MPB}_i \leqslant 1/p$, but $1/p$ is the smallest possible value, so $\mathrm{MPB}_i = 1/p$. For an agent $i \in R$, we had $\mathrm{MPB}_i = 1$ at time $t_{k,a}$ by (H7). Since price reductions can only increase the value of MPB_i and 1 is the highest possible pain-per-buck ratio, we also have $\mathrm{MPB}_i = 1$ at time $t_{k+1,a}$. This proves (H7). □

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 [9] and strictly lexicographic utilities [22]. In this section, we generalize these results to the classes of weakly lexicographic and factored personalized bivalued utilities.[4] 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]Both binary and strictly lexicographic utilities are special cases of weakly lexicographic utilities. Binary utilities ($\{0, 1\}$ or $\{0, -1\}$) can also be seen as equivalent to $\{1, m\}$ or $\{-1, -m\}$ utilities via Lemma 2.3, which are factored bivalued utilities.

4.1.1 Ordered Instances. Bouveret and Lemaître [9, 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 ([9]). *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, 17, 19, 20, 24, 25].

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) $v_i(S) \geqslant \mathrm{MMS}_i^n(\mathcal{M})$, and
(2) $\mathrm{MMS}_j^{n-1}(\mathcal{M} \setminus S) \geqslant \mathrm{MMS}_j^n(\mathcal{M})$ 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 is receives her MMS value in $\mathbf{x}$ by (1), and for any other agent j, $v_j(\mathbf{x}'_j) \geqslant \mathrm{MMS}_j^{n-1}(\mathcal{M} \setminus S) \geqslant \mathrm{MMS}_j^n(\mathcal{M})$ 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 $v_i(S) \geqslant \mathrm{MMS}_i^n(\mathcal{M})$ 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.*

Proof. Fix an agent $i' \in \mathcal{N} \setminus \{i\}$. Given the definition of a valid reduction, we only need to show that $\mathrm{MMS}_{i'}^{n-1}(\mathcal{M} \setminus S) \geqslant \mathrm{MMS}_{i'}^n(\mathcal{M})$. Fix a maximin n-partition $P_{i'}$ as in the statement of the lemma, and note that $\mathrm{MMS}_{i'}^n(\mathcal{M}) = \min_{k \in [n]} v_{i'}(S'_k)$. Then, it is sufficient to construct an $(n-1)$-partition of $\mathcal{M} \setminus S$ such that each bundle in this partition is worth at least $\mathrm{MMS}_{i'}^n(\mathcal{M})$ to agent i'.

Goods division: In this case, we construct the desired $(n-1)$-partition of $\mathcal{M} \setminus S$ by starting from the maximin n-partition P' of $\mathcal{M}$ and deleting bundle S'. Note that none of the other bundles contain a good from S because $S \subseteq S'$. Since deleting a bundle can only improve the utility of the agent for the worst bundle, the utility of agent i' for the worst bundle in the new partition is at least her utility for the worst bundle in P', which is $\mathrm{MMS}_{i'}^n(\mathcal{M})$.

Chore division: In this case, we construct the desired $(n-1)$-partition of $\mathcal{M} \setminus S$ as before, by starting from the maximin n-partition P' of $\mathcal{M}$, deleting bundle S', and deleting any chores in $S \setminus S'$ from the remaining bundles. Since $S \supseteq S'$, this is an $(n-1)$-partition of $\mathcal{M} \setminus S$. Further, deleting a bundle and deleting some chores from the remaining bundles can only improve the utility of the agent for the worst bundle. Hence, the utility of agent i' for the worst bundle in the new partition is at least her utility for the worst bundle in P', which is $\text{MMS}_{i'}^n(\mathcal{M})$. □

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 [16, 30].

Algorithm 2 considers the items in a nonincreasing order of their absolute value and greedily assigns them to the bundle with the lowest total absolute value. In the end, the value of the least-valued bundle is the MMS value. This works for both goods division and chore division.

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

```
x ← (x_i = ∅)_{i∈N}                              // x denotes a partial allocation
for r ∈ M in a nonincreasing order of |v(r)| do
    k* ← arg min_{k∈N} |v(x_k)|
    x_{k*} ← x_{k*} ∪ {r}
return 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.*

Proof. Let $\mathbf{x}$ be the partition returned by the algorithm. Without loss of generality, let $r_1, \ldots, r_m$ be the order in which the algorithm considers the items. Then, $v(r_{k+1}) \mid v(r_k)$ for $k \in [m-1]$. Suppose for contradiction that this is not a maximin partition. Among all maximin partitions, choose $\mathbf{x}'$ such that the lowest index ℓ for which $\mathbf{x}$ and $\mathbf{x}'$ differ in the assignment of item r_ℓ is the maximum possible.

Let $\mathbf{y}$ denote the partial allocation of items $r_1, \ldots, r_{\ell-1}$ under both $\mathbf{x}$ and $\mathbf{x}'$. Let i and i' be such that $r_\ell \in \mathbf{x}_i \cap \mathbf{x}'_{i'}$; note that $i \neq i'$. Because the algorithm assigns r_ℓ to bundle i given the partial allocation $\mathbf{y}$, we must have $|v(\mathbf{y}_i)| \leqslant |v(\mathbf{y}_{i'})|$. We consider two cases.

Case 1: Suppose $|v(\mathbf{y}_i)| = |v(\mathbf{y}_{i'})|$. Let $\widehat{\mathbf{x}}$ be a partition obtained by starting from the maximin partition $\mathbf{x}'$, and swapping the items in $\mathbf{x}'_i \setminus \mathbf{y}_i$ and the items in $\mathbf{x}'_{i'} \setminus \mathbf{y}_{i'}$ between bundles i and i'. Note that $v(\widehat{\mathbf{x}}_i) = v(\mathbf{x}'_{i'})$ and $v(\widehat{\mathbf{x}}_{i'}) = v(\mathbf{x}'_i)$. Hence, $\widehat{\mathbf{x}}$ is also a maximin partition. But it matches $\mathbf{x}$ in the assignment of the first ℓ items, which is a contradiction.

Case 2: Suppose $|v(\mathbf{y}_i)| < |v(\mathbf{y}_{i'})|$. Note that because $v(r_\ell) \mid v(r_k)$ for all $k < \ell$, we must have $|v(\mathbf{y}_i)| \leqslant |v(\mathbf{y}_{i'})| - |v(r_\ell)|$. Here, we consider two sub-cases.

- *Case 2a:* Suppose $|v(\mathbf{x}'_i)| < |v(\mathbf{y}_i)| + |v(r_\ell)| \leqslant |v(\mathbf{y}_{i'})|$. Let $\widehat{\mathbf{x}}$ be a partition obtained by starting from the maximin partition $\mathbf{x}'$, and swapping the items in $\mathbf{x}'_i \setminus \mathbf{y}_i$ and the item r_ℓ between bundles i and i'. Note that $|v(\widehat{\mathbf{x}}_i)| = |v(\mathbf{y}_i)| + |v(r_\ell)| > |v(\mathbf{x}'_i)|$ and $|v(\widehat{\mathbf{x}}_{i'})| \geqslant |v(\mathbf{y}_{i'})| > |v(\mathbf{x}'_i)|$. Hence, the minimum absolute value across bundles weakly increases when moving from $\mathbf{x}'$ to $\widehat{\mathbf{x}}$. Similarly, note that $|v(\widehat{\mathbf{x}}_i)| = |v(\mathbf{y}_i)| + |v(r_\ell)| \leqslant |v(\mathbf{y}_{i'})| \leqslant |v(\mathbf{x}'_{i'})|$ and $|v(\widehat{\mathbf{x}}_{i'})| < |v(\mathbf{y}_{i'})| + |v(r_\ell)| \leqslant |v(\mathbf{x}'_{i'})|$. Hence, the maximum absolute value across bundles weakly decreases when moving from $\mathbf{x}'$ to $\widehat{\mathbf{x}}$. This implies that $\widehat{\mathbf{x}}$ is also a maximin partition. However, $\widehat{\mathbf{x}}$ matches $\mathbf{x}$ in the assignment of the first ℓ items, which is a contradiction.

- *Case 2b:* Suppose $|v(\mathbf{x}'_i)| \geqslant |v(\mathbf{y}_i)|+|v(r_\ell)|$. Suppose $\mathbf{x}'_i \setminus \mathbf{y}_i = \{r_{k_1}, r_{k_2}, \ldots\}$ where $k_1 < k_2 < \ldots$. Let t be the smallest index such that $|v(\mathbf{y}_i \cup \{r_{k_1}, \ldots, r_{k_t}\})| \geqslant |v(\mathbf{y}_i)|+|v(r_\ell)|$. Note that $|v(\mathbf{y}_i \cup \{r_{k_1}, \ldots, r_{k_{t-1}}\})| < |v(\mathbf{y}_i)| + |v(r_\ell)|$. Further, since $v(r_{k_t}) \mid v(r)$ for all $r \in \mathbf{y}_i \cup \{r_{k_1}, \ldots, r_{k_{t-1}}\}$, we must have $|v(\mathbf{y}_i \cup \{r_{k_1}, \ldots, r_{k_{t-1}}\})| \leqslant |v(\mathbf{y}_i)| + |v(r_\ell)| - |v(r_{k_t})|$. Hence, it must be the case that $|v(\mathbf{y}_i \cup \{r_{k_1}, \ldots, r_{k_t}\})| = |v(\mathbf{y}_i)| + |v(r_\ell)|$, i.e., $|v(\{r_{k_1}, \ldots, r_{k_t}\})| = |v(r_\ell)|$. In this case, swapping the set of items $\{r_{k_1}, \ldots, r_{k_t}\}$ with the item r_ℓ between bundles i and i' in $\widehat{\mathbf{x}}$ produces another maximin partition which matches $\mathbf{x}$ in the assignment of the first ℓ items, which is a contradiction.

This completes the proof. □

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

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 $|v_i(1)| \geqslant \ldots \geqslant |v_i(m)|$ 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.

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 4.7. *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 $S = \{1, n+1, \ldots, kn+1\}$, where $k = \lfloor (C_i - 1)/n \rfloor$.*

Proof. Note that because C_i is the smallest bad cut of agent i, we have that for each $k' \in [k]$, agent i has equal utility for all items in $[(k'-1)n+1, k'n]$. Consider how Algorithm 2 constructs a maximin n-partition for agent i given v_i. First, for each $k' \in [k]$, it divides items in $[(k'-1)n+1, k'n]$ equally between the bundles (one to each). Then, it assigns items $[kn+1, C_i]$ to $C_i - kn$ many bundles, again one to each. Note that $C_i - kn \geqslant 1$ by the definition of k. Hence, the first bundle is precisely S.

Note that these $C_i - kn$ bundles receive an extra item $[kn+1, C_i]$ from compared to the remaining $n - (C_i - kn)$ bundles. Further, because v_i is weakly lexicographic and C_i is a bad cut, this item has more absolute value than all items indexed greater than C_i combined. Thus, Algorithm 2 divides the remaining items $[C_i + 1, m]$ among the remaining $n - (C_i - kn)$ bundles. In particular, bundle 1 does not receive any of these items. Hence, in the end, bundle 1 contains exactly the set of items S, as needed. □

Next, we show that if we choose an agent i with the minimum or maximum C_i and the corresponding S from Lemma 4.7, the pair (i, S) forms a valid reduction. Note that this valid reduction can trivially be found in polynomial time.

Lemma 4.8. *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 $S = \{1, n+1, \ldots, kn+1\}$, where $k = \lfloor (C_i - 1)/n \rfloor$.*

Proof. By Lemma 4.7, we have S is one of the bundles in a maximin n-partition of agent i; hence, $v_i(S) \geqslant \mathrm{MMS}_i$.

Consider any other agent i', define $k' = \lfloor (C_{i'} - 1)/n \rfloor$, and let $S' = \{1, n+1, \ldots, k'n+1\}$. Then, by Lemma 4.7, S' is one of the bundles in a maximin n-partition of agent i'. Further, by our choice of agent i, we have $k \leqslant k'$ (thus, $S \subseteq S'$) for goods division, and $k \geqslant k'$ (thus, $S \supseteq S'$) for chore division. It is easy to see that S and S' satisfy the necessary conditions from Lemma 4.4. Hence, (i, S) is a valid reduction. □

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 4.9 (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 $T_i^{\text{idle}} = \min\{p_i \cdot AC_i, m - C_i\}$.

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 4.10. *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 $S = \{1, n+1, \ldots, kn+1\}$, where $k = \lfloor \frac{m - \max\{T_i^{\text{idle}} - AC_i, 0\} - 1}{n} \rfloor$.*

Proof. Fix an agent $i \in \mathcal{N}$. First, if agent i does not have a bad cut, then $T_i^{\text{idle}} = AC_i = 0$, so $m - \max\{(T_i^{\text{idle}} - AC_i), 0\} = m$. In this case, Algorithm 2 simply allocates all items between n bundles in a round robin fashion, so S coincides with the first bundle.

Next, suppose agent i has a bad cut $C_i = k'n + r$, where $r \in [n-1]$. In this case, the algorithm divides the first C_i items of absolute value p_i between all n bundles in a round robin fashion, after which point the first bundle has items $\{1, n+1, \ldots, k'n+1\}$. After this, Algorithm 2 enters the second phase of allocating items of absolute value 1 between the active bundles in a round robin fashion, which runs for T_i^{idle} steps.

If $T_i^{\text{idle}} \leqslant AC_i$, then there must be at most AC_i items left after the bad cut C_i. Thus, $k'n + 1 \leqslant C_i \leqslant m \leqslant C_i + AC_i = (k'+1)n$. This implies that $k = \lfloor (m-1)/n \rfloor = k'$. Hence, S is precisely the first bundle produced by Algorithm 2.

Finally, suppose $T_i^{\text{idle}} \geqslant AC_i$. Since all items in $[C_i + 1, m]$ have absolute value 1, we can do the following during the second phase of Algorithm 2: first allocate items $\{C_i + 1, \ldots, C_i + AC_i\}$ to the active bundles, one each, and then allocate items $\{m - (T_i^{\text{idle}} - AC_i) + 1, \ldots, m\}$ to the active bundles in a round robin fashion. Items $\{C_i + AC_i + 1, \ldots, m - (T_i^{\text{idle}} - AC_i)\}$ (if any) are reserved for the third phase in which items with absolute value 1 need to be divided in a round robin fashion between all n bundles. This change can be interpreted as running Algorithm 2 with a different tie-breaking among items with absolute value 1. Hence, by Lemma 4.5, this still produces a maximin partition. Under this partition, the items allocated to bundle 1 (which is necessarily *not* an active bundle) are those that would be allocated if we allocate items $\{1, \ldots, m - (T_i^{\text{idle}} - AC_i)\}$ in a round robin fashion, i.e., $S = \{1, n+1, \ldots, kn+1\}$, where $k = \lfloor (m - (T_i^{\text{idle}} - AC_i) - 1)/n \rfloor$, as needed. □

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 4.10 yields a valid reduction (i, S).

Lemma 4.11. *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* $k = \lfloor (m - \max\{(T_i^{\text{idle}} - AC_i), 0\} - 1)/n \rfloor$.

Proof. By Lemma 4.10, we know that S is one of the bundles in a maximin n-partition of agent i; hence, $v_i(S) \geqslant \text{MMS}_i$.

For any other agent i', define $S' = \{1, n+1, \ldots, k'n+1\}$, where $k' = \lfloor (m - \max\{(T_{i'}^{\text{idle}} - AC_{i'}), 0\} - 1)/n \rfloor$. Then, by Lemma 4.10, S' is one of the bundles in a maximin n-partition of agent i'. Further, due to our choice of i and using the same reasoning as used in the proof of Lemma 4.8, we have that $S \subseteq S'$ for goods division and $S \supseteq S'$ for chore division, which satisfies the condition of Lemma 4.4. □

4.5 Achieving Pareto Optimal MMS Allocations

In this section, we show that for weakly lexicographic as well as factor 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 4.12. *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 each* $t \in \{2, \dots, k+1\}$ *is a Pareto improvement over* $\mathbf{x}$*.*

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

Given bivalued utilities with values $0 < a < b$ (goods division) or $0 > a > b$ (chore division), an allocation $\mathbf{y}$, and an agent i, define $\mathbf{y}_i^+$ (resp., $\mathbf{y}_i^-$) as the set of items in $\mathbf{y}_i$ for which agent i has value b (resp., a).

Proof. The goods division case is proved by Aziz et al. [4]. Let us focus on chore division. First, let us establish the existence of the special Pareto improvement $\mathbf{x}^*$ in case $\mathbf{x}$ is not PO. Note that if we establish the existence of the desired cycles of agents and items, then the first two properties of these cycles claimed in the lemma imply that the reallocation that yields $\mathbf{x}^*$ makes each agent weakly better off and some agent strictly better off, i.e., that it is a Pareto improvement.

Chore division, weakly lexicographic utilities: Let $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$ be a chore division instance with weakly lexicographic utilities and $\mathbf{x}$ be an allocation that is not Pareto optimal. Among all Pareto improvements, let $\widehat{\mathbf{x}}$ be the one that is closest to $\mathbf{x}$ in that it minimizes $| \cup_{i \in \mathcal{N}} \widehat{\mathbf{x}}_i \setminus \mathbf{x}_i|$.

Consider an agent i_1 who is strictly better off under $\widehat{\mathbf{x}}$ than under $\mathbf{x}$; such an agent must exist in a Pareto improvement. Then, there must exist a chore $c_1 \in \mathbf{x}_{i_1} \setminus \widehat{\mathbf{x}}_{i_1}$. Let $i_2 \neq i_1$ be such that $c_1 \in \widehat{\mathbf{x}}_{i_2}$. Because agent i_2 receives chore c_1 under $\widehat{\mathbf{x}}$, and she is weakly better off under $\widehat{\mathbf{x}}$ than under $\mathbf{x}$, and her utility function is weakly lexicographic, she must have lost a chore $c_2 \in \mathbf{x}_{i_2} \setminus \widehat{\mathbf{x}}_{i_2}$ with $v_{i_2}(c_1) \geqslant v_{i_2}(c_2)$.

More generally, for $k \geqslant 2$, suppose we obtain a sequence of chores $(c_1, \dots, c_{k-1})$ and a sequence of agents $(i_1, \dots, i_k)$ such that chore c_t is transferred from agent i_t to agent i_{t+1} for each $t \in [k-1]$. Then, because agent i_k receives chore c_{k-1} under $\widehat{\mathbf{x}}$, and she is weakly better off under $\widehat{\mathbf{x}}$ than under $\mathbf{x}$, and her utility function is weakly lexicographic, she must have lost a chore $c_k \in \mathbf{x}_{i_k} \setminus \widehat{\mathbf{x}}_{i_k}$ with $v_{i_k}(c_{k-1}) \geqslant v_{i_k}(c_k)$ to another agent i_{k+1}, and the sequence continues. Since the number of agents is finite, this process must run into a cycle where $i_{k+1} = i_\ell$ for some $\ell < k+1$.

If $\ell > 1$, then note that the allocation obtained by starting from $\widehat{\mathbf{x}}$ and reassigning chore c_t back to agent i_t for each $t \in \{\ell, \ell+1, \dots, k\}$ is still a Pareto improvement over $\mathbf{x}$ and is closer to $\mathbf{x}$, which contradicts the definition of $\widehat{\mathbf{x}}$. Hence, we must have $\ell = 1$, in which case the cycles of agents and chores we have constructed are the cycles sought in the lemma (with c_{k+1} defined as c_1).

Chore division, bivalued utilities: Let $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$ be a chore division instance with bivalued utilities and $\mathbf{x}$ be an allocation that is not Pareto optimal. Among all Pareto improvements, choose $\widehat{\mathbf{x}}$ to be the closest to $\mathbf{x}$ in the sense of minimizing $| \cup_{i \in \mathcal{N}} \widehat{\mathbf{x}}_i \setminus \mathbf{x}_i|$.

First, we show that there is no *clear winner* i in $\widehat{\mathbf{x}}$ for whom $\widehat{\mathbf{x}}_i \subsetneq \mathbf{x}_i$. If this were the case, we could take a chore $c \in \mathbf{x}_i \setminus \widehat{\mathbf{x}}_i$ and give it back to agent i in $\widehat{\mathbf{x}}$. The resulting allocation would still be a Pareto improvement over $\mathbf{x}$ (agent i is still weakly better, and the agent who gives c back must now be strictly better), and it would be closer to $\mathbf{x}$, which contradicts the definition of $\widehat{\mathbf{x}}$.

Next, we show that because $\widehat{\mathbf{x}}$ is a Pareto improvement over $\mathbf{x}$, we must have $|\cup_{i \in \mathcal{N}} \widehat{\mathbf{x}}_i^+| < |\cup_{i \in \mathcal{N}} \mathbf{x}_i^+|$. This is because for any allocation $\mathbf{y}$, the social welfare (the sum of utilities of agents) under $\mathbf{y}$ is

$$| \cup_{i \in \mathcal{N}} \mathbf{y}_i^+| \cdot b + | \cup_{i \in \mathcal{N}} \mathbf{y}_i^-| \cdot a = | \cup_{i \in \mathcal{N}} \mathbf{y}_i^+| \cdot (b-a) + m \cdot a.$$

The desired inequality follows from observing that the social welfare under $\widehat{\mathbf{x}}$ must be strictly more than the social welfare under $\mathbf{x}$ and $b - a < 0$.

Consider a chore $c_1 \in \cup_{i \in \mathcal{N}} \mathbf{x}_i^+ \setminus \cup_{i \in \mathcal{N}} \widehat{\mathbf{x}}_i^+$. Suppose $c_1 \in \mathbf{x}_{i_1}^+ \cap \widehat{\mathbf{x}}_{i_2}^-$ for some $i_2 \neq i_1$. We can represent this as $i_1 \underset{\mathbf{x}}{\overset{b}{\rightarrow}} c_1 \underset{\widehat{\mathbf{x}}}{\overset{a}{\rightarrow}} i_2$, where, for an arrow connecting agent i with chore c, the entry above indicates $v_i(c)$ while the entry below indicates the allocation in which c is allocated to i. Consider extending this chain as much as possible by adding alternating $\xrightarrow[\mathbf{x}]{a}$ and $\xrightarrow[\widehat{\mathbf{x}}]{a}$ edges to obtain $i_1 \xrightarrow[\mathbf{x}]{b} c_1 \xrightarrow[\widehat{\mathbf{x}}]{a} i_2 \dots c_{t-1} \xrightarrow[\widehat{\mathbf{x}}]{a} i_t$. There are two possibilities: either the chain stops at agent i_t for some $t \geqslant 2$ (and we are unable to extend it further), or an agent repeats at some point (i.e., $i_t = i_\ell$ for some $\ell < t$).

Case 1: the chain stops at agent i_t. First, suppose $\mathbf{x}^+_{i_t} \neq \emptyset$. Consider any chore $\widehat{c_t} \in \mathbf{x}^+_{i_t}$. Consider the allocation obtained by starting from $\mathbf{x}$ and cyclically shifting chores as follows: chore c_k is moved to agent i_{k+1} for $k \in [t-1]$, and chore $\widehat{c_t}$ is moved to agent i_1. Note that agent i_1 loses a b-valued chore and gains a chore, agents i_2 through i_{t-1} each lose an a-valued chore and gain an a-valued chore, and agent i_t loses a b-valued chore and gains an a-valued chore. Thus, this is the kind of cycle sought in the lemma.

Next, suppose $\mathbf{x}^+_{i_t} = \emptyset$. Because $\widehat{\mathbf{x}}$ is a Pareto improvement over $\mathbf{x}$ in which agent i_t gains a new chore c_{t-1}, she must have also lost at least one chore. Pick $c_t \in \mathbf{x}^-_{i_t} \setminus \widehat{\mathbf{x}}_{i_t}$. Let $c_t \in \widehat{\mathbf{x}}_{i_{t+1}}$. If $c_t \in \widehat{\mathbf{x}}^-_{i_{t+1}}$, then the chain could have continued. Hence, it must be the case that $c_t \in \widehat{\mathbf{x}}^+_{i_{t+1}}$. In this case, consider the allocation obtained by starting from $\widehat{\mathbf{x}}$ and exchanging chores c_{t-1} and c_t between agents i_t and i_{t+1}. Note that the utility to agent i_t does not change because she loses an a-valued chore and gains an a-valued chore, and agent i_{t+1} is weakly better because she loses a b-valued chore and gains a chore. Hence, the resulting allocation is still a Pareto improvement over $\mathbf{x}$. However, it is also closer to $\mathbf{x}$ than $\widehat{\mathbf{x}}$ is, because we give chore c_t back to agent i_t during the exchange. This contradicts the definition of $\widehat{\mathbf{x}}$.

Case 2: $i_t = i_\ell$ for some $\ell < t$. First, suppose $\ell = 1$. Then, consider the allocation obtained by starting from $\mathbf{x}$ and cyclically shifting chores as follows: chore c_k is moved to agent i_{k+1} for $k \in [t-1]$. Note that agent i_1 loses a b-valued chore and gains an a-valued chore, and agents i_2 through i_{t-1} each lose an a-valued chore and gain an a-valued chore. Thus, this is the kind of cycle sought in the lemma.

Finally, suppose $\ell \neq 1$. In this case, consider the allocation obtained by starting from $\widehat{\mathbf{x}}$ and cyclically shifting the chores back as follows: chore c_k is moved back to agent i_k for $k \in \{\ell, \ell+1, \dots, t-1\}$. Compared to $\widehat{\mathbf{x}}$, agents i_ℓ through i_{t-1} each lose an a-valued chore and gain an a-valued chore. Hence, the resulting allocation is still a Pareto improvement over $\mathbf{x}$, and it is closer to $\mathbf{x}$ than $\widehat{\mathbf{x}}$ is, which contradicts the definition of $\widehat{\mathbf{x}}$.

Efficient computation: Finally, for finding the kind of cycles sought in the lemma, we can use the same method that Aziz et al. [4] use for weakly lexicographic utilities. We can create a directed graph with the items as the nodes, and add an edge (r, r') whenever $v_i(r') \geqslant v_i(r)$ for the agent i who holds item r under $\mathbf{x}$. We call this edge strict if $v_i(r') > v_i(r)$. Then, the problem reduces to testing the existence of a cycle in this graph with at least one strict edge (and finding it if it exists). This can be done efficiently by considering each strict edge (r, r'), and trying to find a path in the graph from r' to r. If a cycle is found, the desired Pareto improvement $\mathbf{x}^*$ can be computed efficiently by reallocating items along the cycle. □

The next lemma shows that starting from any allocation, if we repeatedly find a Pareto improvement by invoking Lemma 4.12, then we arrive at a Pareto optimal allocation in at most a polynomial number of steps.

LEMMA 4.13. *Let $\mathbf{x}^0$ be an allocation in a goods division or chore division instance with weakly lexicographic or bivalued utilities. Let $(\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \dots)$ 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 4.12. Then, the chain terminates at a Pareto optimal allocation in at most a polynomial number of steps.*

Proof. For bivalued utilities, note that the Pareto improvement $\mathbf{x}^*$ identified in Lemma 4.12 strictly increases (resp., reduces) the number of goods (resp., chores) allocated to agents who value it at b. Since this value is between 0 and m, the chain must end in at most m steps.

Next, consider an instance $I = (\mathcal{N}, \mathcal{M}, \mathbf{v})$ with weakly lexicographic utilities. Let us define a quantity $h(i, r)$ for every agent i and item r: if $(L_1, \dots, L_k)$ is the partition of $\mathcal{M}$ under the weakly lexicographic utility function v_i of agent i as in Definition 2.2 and $r \in L_t$, then we set $h(i, r) = t$. For an allocation $\mathbf{y}$, define the potential function $\phi(\mathbf{y}) = \sum_{i \in \mathcal{N}} \sum_{r \in \mathbf{y}_i} h(i, r)$. Note that $m \leqslant \phi(\mathbf{y}) \leqslant m^2$. We show that in case of goods division (resp., chore division), every Pareto improvement in the chain strictly decreases (resp., increases) the potential. This implies that the chain must terminate in $O(m^2)$ steps.

Consider any Pareto improvement from $\mathbf{x}$ to $\mathbf{x}^*$ obtained by a cycle of items $(r_1, \dots, r_k, r_{k+1} = r_1)$ as in Lemma 4.12. For any agent i, note that for every item r_t that she loses, she gains a unique item r_{t-1} with $v_i(r_{t-1}) \geqslant v_i(r_t)$, which implies $h(i, r_{t-1}) \leqslant h(i, r_t)$ for goods division and the opposite inequality for chore division. Thus, $\sum_{r \in \mathbf{x}_i^*} h(i, r) \leqslant \sum_{r \in \mathbf{x}_i} h(i, r)$ for goods division and the opposite inequality holds for chore division. Further, because the Pareto improvement strictly improves the utility of some agent, the inequality for that agent is strict. Hence, the Pareto improvement strictly decreases (resp., increases) the potential value for goods division (resp., chore division), as desired. □

In Lemma 4.13, 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 4.14. *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 [6], 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 [18] 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.

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] 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.

[7] 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.

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

[9] 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.

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

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

[12] 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.

[13] 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.

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

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

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

[17] 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.

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.* 630–631.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

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

[30] Gerhard J Woeginger. 1997. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters* 20, 4 (1997), 149–154.

APPENDIX

A PROOF OF LEMMA 3.8

In this section, we prove the two useful properties of Phase 2b claimed in Lemma 3.8. These properties hold for general additive utilities, as shown in Barman et al. [6]. In Phase 2b, we use a relaxed condition in line 18 compared to the Phase 2 described in Barman et al. [6]. The difference is that for the shortest MPB alternating path $\mathrm{ls} \xleftarrow{c_1} i_1 \xleftarrow{c_2} \cdots \xleftarrow{c_\ell} i_\ell$, instead of making a transfer when $\mathbf{p}(\mathbf{x}_{i_\ell}) - \mathbf{p}(c_\ell) > \mathbf{p}(\mathbf{x}_{\mathrm{ls}})$ (referred to as a "path violator"), we do a transfer when $\mathbf{p}_{\text{up to }1}(\mathbf{x}_{i_\ell}) > \mathbf{p}(\mathbf{x}_{\mathrm{ls}})$ (referred to as a "violator"). Note that $\mathbf{p}(\mathbf{x}_{i_\ell}) - \mathbf{p}(c_\ell) \leqslant \mathbf{p}_{\text{up to }1}(\mathbf{x}_{i_\ell})$. Therefore, if our Phase2b makes a transfer then the original Phase 2 in [6] also does that.

One observation is that the minimum spending value never decreases over the run of a Phase 2b. This is Lemma 3.8 (2).

Lemma A.1. *During a run of Phase 2b, the minimum spending, i.e.* $\min_{i\in\mathcal{N}} \mathbf{p}(\mathbf{x}_i)$, *never decreases.*

Proof. It is sufficient to show the minimum spending does not decrease after each single transfer of Phase 2b. Suppose we transfer a chore c from agent i to agent j. Let ls be the least spender before the transfer. Let $\mathbf{x}$ and $\mathbf{x}'$ denote the allocations before and after the transfer respectively. Right before the transfer, i must have been a violator to ls, then $\mathbf{p}(\mathbf{x}_{\mathrm{ls}}) < \mathbf{p}_{\text{up to }1}(\mathbf{x}_i) \leqslant \mathbf{p}(\mathbf{x}_i) - \mathbf{p}(c) = \mathbf{p}(\mathbf{x}'_i)$. That is, the spending of i after the transfer is strictly larger than the minimum spending before the transfer. Furthermore, spending of j has increased by $\mathbf{p}(c)$, and other agents have the same bundle (and spending) as they had before the transfer. In conclusion, $\min_{i'\in\mathcal{N}} \mathbf{p}(\mathbf{x}_{i'}) \leqslant \min_{i'\in\mathcal{N}} \mathbf{p}(\mathbf{x}'_{\mathrm{ls}})$. □

Next, we show that Phase 2b must terminate after at most $\mathrm{poly}(n, m, \max_{i\in\mathcal{N}}|\mathcal{U}_i|)$ steps, where U_i is the set of all different utilities agent i has for all subsets of items. The proof follows from the next two lemmas.

Lemma A.2 (Lemma 13, Barman et al. [6]). *After* $\mathrm{poly}(n, m)$ *steps in Phase 2b, either the identity of the least spender changes or a Phase 3 happens.*

Proof. At time t, let LS be the set of agents with the minimum spending, and for all agents $i \in \mathcal{N}$ define

$$\mathrm{level}(i,t) := \begin{cases} \ell, & \text{if } \exists\, \mathrm{ls} \in LS\colon\ \mathrm{ls} \leftsquigarrow i, \text{ and } \ell \text{ is the length of the shortest of such paths,} \\ n, & \text{if } \nexists\, \mathrm{ls} \in LS\colon\ \mathrm{ls} \leftsquigarrow i. \end{cases}$$

Furthermore, let $G(i,t)$ be the set of chores $c \in \mathbf{x}_i^t$ such that there exists an MPB alternating path $\mathrm{ls} \leftarrow \cdots \leftarrow i' \xleftarrow{c} i$ where the last edge uses chore c. For agents i where $\mathrm{level}(i,t) = n$, $G(i,t) = \emptyset$.

Now, define the potential function $\phi(t)$ as follows,

$$\phi(t) = \sum_{i\in\mathcal{N}} m \cdot (n - \mathrm{level}(i,t)) + |G_{i,t}|.$$

Note that ϕ is always integral and positive, $\sum_{i\in\mathcal{N}}|G_{i,t}| \leqslant m$, and $\sum_{i\in\mathcal{N}} m \cdot (n - \mathrm{level}(i,t)) \leqslant mn^2$. Then, to show the lemma holds, it suffices to prove the potential function strictly decreases after each transfer. Therefore, Phase 2b terminates after $O(mn^2)$ steps.

Suppose we transfer the chore c from agent i_ℓ to agent $i_{\ell-1}$.

Agents at level $\ell - 2$ do not consider c their MPB, because otherwise the shortest path to i_ℓ would be of length $\ell - 1$. Therefore, after $i_{\ell-1}$ receives c, $\mathrm{level}(i_{\ell-1}, t+1) = \mathrm{level}(i_{\ell-1}, t)$ and $G(i_{\ell-1}, t+1) = G(i_{\ell-1}, t)$, and ϕ does not change for the terms related to $i_{\ell-1}$.

Similarly, other agents $i' \in \mathcal{N} \setminus \{i_\ell, i_{\ell-1}\}$ cannot move to lower levels after this transfer.

For i_ℓ, either there exists another chore in $G_{i,t}$ which keeps her in level ℓ, or she moves to a strictly higher level (or possibly $\mathrm{level}(i_\ell, t+1) = n$). In either case, we can show ϕ strictly decreases.

If $\mathrm{level}(t, i_\ell) < \mathrm{level}(t+1, i_\ell)$, then any change in $\sum_{i\in\mathcal{N}}|G_{i,t}|$ will be cancelled out by the decrease in $m \cdot (n - \mathrm{level}(i_\ell, t))$ due to the lexicographical weighting.

If $\mathrm{level}(t, i_\ell) = \mathrm{level}(t+1, i_\ell)$, then for other agents $i' \in \mathcal{N} \setminus \{i_\ell, i_{\ell-1}\}$, $G_{i',t}$ does not change. However, $|G_{i_\ell,t+1}| = |G_{i_\ell,t}| - 1$. Thus, f decreases by at least one after each transfer in Phase 2b. □

Lemma A.3. *During a continuous run of Phase 2b, if agent i ceases being the least spender after time t, and becomes the least spender again at some time $t' > t$, then her utility have must have decreased, i.e. $v_i(\mathbf{x}_i^{t'}) < v_i(\mathbf{x}_i^t)$.*

Proof. When agent i ceases being the least spender, she must have beenshe must have received a chore c at time t. That is $\mathbf{p}(\mathbf{x}_i^t) = \min_{i'\in\mathcal{N}} \mathbf{p}(\mathbf{x}_{i'}^t)$, and $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t \cup \{c\}$. First, suppose $\mathbf{x}_i^{t+1} \subsetneq \mathbf{x}_i^{t'}$, i.e. i did not give away any chores from $t+1$ to t', then $v_i(\mathbf{x}_i^{t'}) \leqslant v_i(\mathbf{x}_i^{t+1}) < v_i(\mathbf{x}_i^t)$.

Now, assume i has given away at least one chore from $t+1$ to t'. Let t_ℓ be the last time she gave away an item. Suppose that item is c'. At t_ℓ, i must have been a violator to the least spender, say agent ls, then

$$\mathbf{p}(\mathbf{x}_{\mathrm{ls}}^{t_\ell}) < \mathbf{p}_{\text{up to 1}}(\mathbf{x}_i^{t_\ell}) \leqslant \mathbf{p}(\mathbf{x}_i^{t_\ell}) - \mathbf{p}(c') = \mathbf{p}(\mathbf{x}_i^{t_\ell+1}).$$

Furthermore, as i was the least spender at t and the minimum spending has not decreased by Lemma A.1, $\mathbf{p}(\mathbf{x}_i^t) \leqslant \mathbf{p}(\mathbf{x}_{\mathrm{ls}}^{t_\ell})$. Putting these together, we conclude that $\mathbf{p}(\mathbf{x}_i^t) < \mathbf{p}(\mathbf{x}_i^{t_\ell+1})$. Since this was the last time i gave away a chore, her spending could not go any lower. Thus, $\mathbf{p}(\mathbf{x}_i^t) < \mathbf{p}(\mathbf{x}_i^{t'})$. There were no price changes between t and t', then MPB_i has remained the same, and $|v_i(\mathbf{x}_i^t)| = \mathrm{MPB}_i \cdot \mathbf{p}(\mathbf{x}_i^t) < \mathrm{MPB}_i \cdot \mathbf{p}(\mathbf{x}_i^{t'}) = |v_i(\mathbf{x}_i^{t'})|$ which completes the proof. □

With the two lemmas above, we can prove an upper bound on the running time of Phase 2b.

Lemma A.4. *Phase 2b of Algorithm 1 should terminate after at most* $\mathrm{poly}(n, m, \max_{i\in\mathcal{N}}|\mathcal{U}_i|)$ *time, where $\mathcal{U}_i = \{v_i(S) \mid \forall S \subseteq \mathcal{M}\}$ is the set of all different utilities agent i can obtain.*

Proof. By Lemma A.3, the number of times an agent ceases being among the least spenders is bounded by the number of different utilities she can have, i.e. $\max_{i\in\mathcal{N}}|\mathcal{U}_i|$. Moreover, by Lemma A.2, after $\mathrm{poly}(n, m)$ time the identity of the least spender must change or we turn to a Phase 3. Therefore, a continuous Phase 2 can run for at most $\mathrm{poly}(n, m, \max_{i\in\mathcal{N}}|\mathcal{U}_i|)$ □

As a corollary of Lemma A.4 applied to the bivalued chores case, and due to the fact that $|\mathcal{U}_i| \leqslant m^2$ (fix the number of -1's and $-p$'s in the bundle), Phase 2b terminates in $\mathrm{poly}(n, m)$ time. Therefore, we have proved part (1) of Lemma 3.8 as Lemma A.4 and part (2) of Lemma 3.8 as Lemma A.1.