Journal of Artificial Intelligence Research 80 (2024) 1-25 Submitted 09/2023; published 05/2024

Computing Pareto-Optimal and Almost Envy-Free Allocations of Indivisible Goods

Jugal Garg JUGAL@ILLINOIS.EDU
University of Illinois, Urbana-Champaign

Aniket Murhekar ANIKET2@ILLINOIS.EDU
University of Illinois, Urbana-Champaign

Abstract

We study the problem of fair and efficient allocation of a set of indivisible goods to agents with additive valuations using the popular fairness notions of envy-freeness up to one good (EF1) and equitability up to one good (EQ1) in conjunction with Pareto-optimality (PO). There exists a pseudo-polynomial time algorithm to compute an EF1+PO allocation and a non-constructive proof of the existence of allocations that are both EF1 and fractionally Pareto-optimal (fPO), which is a stronger notion than PO. We present a pseudo-polynomial time algorithm to compute an EF1+fPO allocation, thereby improving the earlier results. Our techniques also enable us to show that an EQ1+fPO allocation always exists when the values are positive and that it can be computed in pseudo-polynomial time.

We also consider the class of k-ary instances where k is a constant, i.e., each agent has at most k different values for the goods. For such instances, we show that an EF1+fPO allocation can be computed in strongly polynomial time. When all values are positive, we show that an EQ1+fPO allocation for such instances can be computed in strongly polynomial time. Next, we consider instances where the number of agents is constant and show that an EF1+PO (likewise, an EQ1+PO) allocation can be computed in polynomial time. These results significantly extend the polynomial-time computability beyond the known cases of binary or identical valuations.

We also design a polynomial-time algorithm that computes a Nash welfare maximizing allocation when there are constantly many agents with constant many different values for the goods. Finally, on the complexity side, we show that the problem of computing an EF1+fPO allocation lies in the complexity class PLS.

1. Introduction

The problem of fair division was formally introduced by Steinhaus (1949) and has since been extensively studied in various fields, including economics and computer science (Brams & Taylor, 1996; Moulin, 2004). It concerns allocating resources to agents in a *fair* and *efficient* manner and has various practical applications such as rent division, division of inheritance, course allocation, and government auctions. Much of earlier work has focused on *divisible* goods, which agents can share. In this setting, a prominent fairness notion is *envy-freeness* (Foley, 1967; Varian, 1974), which requires that every agent prefer their own bundle of goods to that of any other. On the other hand, when the goods are *indivisible*, envy-free allocations need not even exist, e.g., in the simple case of one good and two agents. Other classical notions of fairness, like *equitability* and *proportionality*, may also be impossible to satisfy when goods are indivisible. However, fair division of indivisible goods remains an important problem since goods cannot always be shared and because it models several practical scenarios such as a course allocation (Othman, Sandholm, & Budish, 2010). We refer the reader to the recent surveys (Walsh, 2020; Amanatidis, Aziz, Birmpas, Filos-Ratsikas, Li, Moulin, Voudouris, & Wu, 2023) for other applications and recent results.

Since allocations satisfying standard fairness criteria fail to exist in the case of indivisible goods, several weaker fairness notions have been defined. A relaxation of envy-freeness called *envy-freeness up to one good* (EF1) was defined by Budish (2011). An allocation is said to be EF1 if every agent prefers their own bundle over the bundle of any other agent after removing at most one good from the other agent's bundle. When the valuations of the agents for the goods are monotone, EF1 allocations exist and are polynomial time computable (Lipton, Markakis, Mossel, & Saberi, 2004).

The standard notion of economic efficiency is Pareto optimality (PO). An allocation is said to be PO if no other allocation makes an agent better off without making someone else worse off. A natural question is whether EF1 can be achieved with PO under additive valuations, which is the valuation class we focus on in this work. The concept of *Nash welfare* provides a positive answer to this question. The Nash welfare is the geometric mean of the agents' utilities, and the allocation maximizing it achieves a tradeoff between efficiency and fairness. Caragiannis et al. (2016) showed that any maximum Nash welfare (MNW) allocation is EF1 and PO. For the special case of binary additive valuations, the MNW allocation can be computed in polynomial time (Darmann & Schauer, 2014; Barman et al., 2018b; Halpern et al., 2020). However, in general, the problem of computing the MNW allocation is APX-hard (Lee, 2015; Garg, Hoefer, & Mehlhorn, 2017). Moreover, it is not known if approximately Nash-optimal allocations retain the EF1 fairness guarantee, implying that approximation algorithms for MNW allocation, e.g., (Nguyen & Rothe, 2014; Cole & Gkatzelis, 2015) may not be useful for computing an EF1+PO allocation.

By passing this barrier, Barman, Krishnamurthy, and Vaish (2018a) devised a pseudo-polynomial time algorithm that computes an allocation that is both EF1 and PO. They also showed that allocations that are both EF1 and *fractionally* Pareto-optimal (fPO) always exist, where an allocation is said to be fPO if no fractional allocation exists that makes an agent better off without making anyone else worse off. They showed this result via a non-constructive convergence argument used in real analysis and did not provide an algorithm for computing such an allocation. Clearly, fPO is a stronger notion of economic efficiency, so the problem of computing EF1+fPO allocations is important. Another reason to prefer fPO allocations over PO allocations in practice is that the former property admits efficient verification, whereas checking if an allocation is PO is known to be coNP-complete (de Keijzer, Bouveret, Klos, & Zhang, 2009). When a centralized entity is responsible for the allocation, all participants can efficiently verify if an allocation is fPO (and thus PO). However, in general, the same efficient verification is not possible for PO allocations due to the above coNP-hardness.

In this paper, we present a pseudo-polynomial time algorithm that computes an allocation that is EF1+fPO. Not only does this improve the result of Barman et al. (2018a), but it also provides other interesting insights. We consider the class of k-ary instances. i.e., each agent has at most k different (agent-specific) values for the goods. Our analysis shows that an EF1+fPO allocation can be found in polynomial time for k-ary instances when k is a constant. Our result becomes especially interesting because computing the MNW allocation remains NP-hard for such instances (Lee, 2015), even for $k = 3$ (Amanatidis et al., 2020). Further, at present, this is the *only* class apart from binary or identical valuations for which EF1+fPO allocations are polynomial time computable.

While k-ary instances are interesting theoretically to understand the limits of tractability in computing fair and efficient allocations, they are also relevant from a practical perspective. Eliciting agents' values for goods is often tricky, as agents may not be able to assert exactly what values they have for different goods. A simple protocol that the entity in charge of the allocation can do is to ask each agent to "rate" the goods using a few (constantly many) values. Based on these responses, the valuations of the agents can be established.

Our results also extend to the fairness notion of *Equitability up to one good* (EQ1), which is a generalization of the classical fairness notion of equitability. An allocation is said to be EQ1 if the utility an agent gets from her bundle is no less than the utility any other agent gets after removing one specific good from their bundle. Using similar techniques to that of Barman et al. (2018a), a pseudo-polynomial time algorithm to compute an EQ1+PO allocation was developed by Freeman, Sikdar, Vaish, and Xia (2019) when all the values are positive. We show the stronger result that EQ1+fPO allocations always exist for positive-valued instances and can be computed in pseudo-polynomial time. Our techniques also show that for k-ary instances with positive values where k is a constant, an allocation that is EQ1 and fPO can be computed in polynomial time.

We next show that for constant n, an EF1+PO allocation can be computed in time polynomial in the number of goods. This result is significant since the number of agents n is constant in many practical applications. In contrast, computing the MNW allocation remains NP-hard for $n = 2$. Our techniques also show that for constant n, an EQ1+PO allocation can also be computed in polynomial time.

Further, for k-ary instances with constant n and k, we show that many fair division problems, including computing the MNW allocation, have polynomial time complexity. This improves the result of Bliem, Bredereck, and Niedermeier (2016), showing that the EF+PO problem is tractable in this case.

We also make progress on the complexity front. We prove that the problem of computing an EF1+fPO lies in the complexity class Polynomial Local Search (PLS). For this, we carefully analyze our algorithm computing an EF1+fPO allocation and show that it has the structure of a local-search problem. Finally, we remark that our techniques also improve the results of Chakraborty, Igarashi, Suksompong, and Zick (2020) and Freeman, Sikdar, Vaish, and Xia (2020) for the problems of computing weighted-EF1+fPO allocations of goods and EQ1+fPO allocations of *chores*, respectively.

We summarize our results in Table 1. A preliminary version of the present work appeared at AAAI 2021 (Garg & Murhekar, 2021).

1.1 Related Work

Since the fair division literature is too vast, we refer the reader to surveys (Walsh, 2020; Amanatidis et al., 2023) for results on other fairness notions like proportionality. Below, we mention works related to fairness notions considered in this paper.

EF1+PO for goods. Barman et al. (2018a) devised a pseudo-polynomial time algorithm that computes an allocation that is both EF1 and PO. This algorithm runs in time

Instance type	EF1+fPO	EQ1+fPO*
constant n	$\mathsf{poly}(m)$ (Theorem 12)	$\mathsf{poly}(m)$ (Theorem 13)
k-ary with constant k	$\mathsf{poly}(m, n)$ (Theorem 5)	$\mathsf{poly}(m, n)$ (Theorem 9)
general additive	$\mathsf{poly}(n, m, v_{max})$ (Theorem 4)	$\mathsf{poly}(n, m, v_{max})$ (Theorem 8)

Table 1: Best-known algorithm run-times for the EF1+fPO and EQ1+fPO problems classified according to instance type. Here n and m denote the number of agents and goods, respectively, and v_{max} denotes the maximum utility value. Agents have additive valuations. *Results for the EQ1+fPO problem apply to positive instances.

$\mathsf{poly}(n, m, v_{max})$, where n is the number of agents, m is the number of items, and v_{max} is the maximum utility value. Their algorithm first perturbs the values to a desirable form and then computes an EF1 and fPO allocation for the perturbed instance. Their approach is via *integral market-equilibria*, which guarantees fPO at every step. The *spending* of an agent, which is the sum of prices of the goods she owns in the equilibrium, works as a proxy for her utility. The returned allocation is approximately-EF1 and approximately-PO for the original instance, which, for a fine enough approximation, is EF1 and PO. Our algorithm proceeds similarly to their algorithm, with one main difference being that we do not need to consider any approximate instance and can work directly with the given valuations. Our algorithm returns an allocation that is not only PO but is fPO. Another key difference is the *run-time analysis*: while their analysis relies on bounding the number of steps using arguments about *prices*, our analysis is more direct and works with the *values*. This allows us to prove a general result (Theorem 3), a consequence of which is polynomial run-time for k-ary instances with constant k. Directly, such a conclusion cannot be drawn from the analysis of Barman et al. (2018a). Another technical difference is that we raise the prices of multiple components of *least spenders* simultaneously, unlike in Barman et al. (2018a), and our analysis is arguably simpler than theirs. They also showed that there is a non-deterministic algorithm that computes an EF1+fPO allocation since checking if an allocation is fPO can be done efficiently. In contrast, we present a deterministic algorithm computing an EF1+fPO allocation, albeit with worst-case pseudopolynomial run time. Garg and Murhekar (2023) showed that allocations that are EFX (envy-free up to the removal of *any* good) and fPO exist and can be computed in polynomial time for a subclass of 2-ary instances called bivalued instances.

EF1+PO for chores. Recently, the existence of EF1+PO allocations of *chores* has been shown for special classes, although the question of existence in its full generality remains open. The existence and polynomial time computation of an EF1+fPO allocation for chores is known for (i) Bivalued instances, shown by Garg, Murhekar, and Qin (2022) and Ebadian, Peters, and Shah (2022) independently, (ii) two types of chores (Aziz, Lindsay, Ritossa, & Suzuki, 2022), (iii) $n = 3$ agents (Garg, Murhekar, & Qin, 2023), and (iv) three types of agents (Garg, Murhekar, & Qin, 2024).

EQ1+PO. Freeman et al. (2019) presented a pseudo-polynomial time algorithm for computing an EQ1+PO allocation for instances with positive values. Since they consider an approximate instance, too, their algorithm does not achieve the guarantee of fPO. They also show that the leximin solution, i.e., the allocation that maximizes the minimum utility and, subject to this, maximizes the second minimum utility, and so on, is EQX (a stronger requirement than EQ1) and PO for positive utilities. However, a simple reduction from the partition problem shows that computing a leximin solution is intractable. For positive bivalued instances, Garg and Murhekar (2023) showed that an EQX+fPO allocation can be computed in polynomial time. For chores, (Freeman et al., 2020) showed that EQ1+PO allocations can be computed in pseudo-polynomial time.

Other results. Bredereck et al. (2019) presented a framework for fixed-parameter algorithms for many fairness concepts, including EF1+PO, parameterized by $n + v_{max}$. Our results improve their findings and show that the problem has polynomial time complexity for the cases of (i) constant n, (ii) constant number of utility values, (iii) v_{max} bounded by $\mathsf{poly}(m, n)$.

2. Preliminaries

Let $\mathbb{N}^+$ denote the set of positive integers. For $t \in \mathbb{N}^+$, let $[t]$ denote $\{1, \ldots, t\}$.

Problem setting. A *fair division instance* is a tuple (N, M, V), where $N = [n]$ is a set of $n \in \mathbb{N}^+$ agents, $M = [m]$ is the set of $m \in \mathbb{N}^+$ indivisible items, and $V = \{v_1, \ldots, v_n\}$ is a set of utility functions, one for each agent $i \in N$. Each utility function $v_i : M \to \mathbb{N}^+$ is additive and is specified by m numbers $v_{ij} \in \mathbb{N}$, one for each good $j \in M$, denoting the value agent i has for good j. Additivity of the valuation functions implies that for every agent $i \in N$, and $S \subseteq M$, $v_i(S) = \sum_{j \in S} v_{ij}$. Further, we assume that for every good j, there is some agent i such that $v_{ij} > 0$. Otherwise, the good j for which $v_{ij} = 0$ for all i can be assigned to any agent arbitrarily. Note that we can in general work with rational values since they can be scaled to make them integral.

We call a fair division instance (N, M, V) a:

1. *Binary* instance if for all $i \in N$ and $j \in M$, $v_{ij} \in \{0, 1\}$.

2. *k-ary* instance, if $\forall i \in N$, $|\{v_{ij} : j \in M\}| \leq k$.

3. *Positive-valued* instance if $\forall i \in N$, $\forall j \in M$, $v_{ij} > 0$.

Note that the class of k-ary instances generalizes the class of k-valued instances as defined by Amanatidis et al. (2020), in which all the values belong to a k-sized set, whereas we allow each agent to have k different values for the goods.

Allocation. An *allocation* $\mathbf{x}$ of goods to agents is a n-partition of the goods $\mathbf{x}_1, \ldots, \mathbf{x}_n$, where agent i is allotted $\mathbf{x}_i \subseteq M$, and gets a total value of $v_i(\mathbf{x}_i)$. A *fractional allocation* $\mathbf{x} \in [0, 1]^{n \times m}$ is a fractional assignment such that for each good $j \in M$, $\sum_{i \in N} x_{ij} \leq 1$. Here, $x_{ij} \in [0, 1]$ denotes the fraction of good j allotted to agent i.

For an agent $i \in N$, let U_i be the number of different utility values i can get in any allocation. Let $U = \max_{i \in N} U_i$.

Fairness notions. An allocation $\mathbf{x}$ is said to be:

1. *Envy-free up to one good* (EF1) if for all $i, h \in N$, there exists a good $j \in \mathbf{x}_h$ s.t. $v_i(\mathbf{x}_i) \geq v_i(\mathbf{x}_h \setminus \{j\})$.

2. *Equitable up to one good* (EQ1) if for all $i, h \in N$, there exists a good $j \in \mathbf{x}_h$ s.t. $v_i(\mathbf{x}_i) \geq v_h(\mathbf{x}_h \setminus \{j\})$.

Pareto-optimality. An allocation $\mathbf{y}$ dominates an allocation $\mathbf{x}$ if $v_i(\mathbf{y}_i) \geq v_i(\mathbf{x}_i), \forall i$ and there exists h s.t. $v_h(\mathbf{y}_h) > v_h(\mathbf{x}_h)$. An allocation is said to be *Pareto optimal* (PO) if no allocation dominates it. Further, an allocation is said to be fractionally PO (fPO) if no fractional allocation dominates it. Thus, an fPO allocation is PO, but not vice-versa.

We remark that the EF1 and PO/fPO properties are scale-invariant. That is, if an allocation $\mathbf{x}$ is EF1+fPO for an instance I, then $\mathbf{x}$ is also EF1+fPO for an instance I' where the valuation function of any agent i is multiplied by a positive constant λ_i, i.e., $v'_{ij} = \lambda_i \cdot v_{ij}$ for every $i \in N$ and $j \in M$.

Maximum Nash Welfare. The Nash welfare of an allocation $\mathbf{x}$ is given by $\mathsf{NW}(\mathbf{x}) = \big(\Pi_{i\in N} v_i(\mathbf{x}_i)\big)^{1/n}$. An allocation that maximizes the NW is called an MNW allocation.

Fisher markets. A Fisher market or a *market instance* is a tuple (N, M, V, e), where the first three terms are interpreted as before, and $e = \{e_1, \ldots, e_n\}$ is the set of agents' budgets, where $e_i \geq 0$, for each $i \in N$. In this model, goods can be allocated fractionally. Given prices $\mathbf{p} = (p_1, \ldots, p_m)$ of goods, each agent aims to obtain the set of goods that maximizes her total value subject to her budget constraint.

A *market outcome* is a (fractional) allocation $\mathbf{x}$ of the goods to the agents and a set of prices $\mathbf{p}$. The spending of an agent i under the market outcome $(\mathbf{x}, \mathbf{p})$ is given by $\mathbf{p}(\mathbf{x}_i) = \sum_{j\in M} p_j x_{ij}$. For an agent i, we define the *bang-per-buck* ratio α_{ij} of good j as v_{ij}/p_j, and the *maximum bang-per-buck* (MBB) ratio $\alpha_i = \max_j \alpha_{ij}$. We define $\text{MBB}_i = \{j \in M : v_{ij}/p_j = \alpha_i\}$, called the *MBB-set*, to be the set of goods that give MBB to agent i at prices $\mathbf{p}$. A market outcome $(\mathbf{x}, \mathbf{p})$ is said to be *'on MBB'* if for all agents i and goods j, $x_{ij} > 0$ implies $j \in \text{MBB}_i$. For integral $\mathbf{x}$, this means $\mathbf{x}_i \subseteq \text{MBB}_i$ for all $i \in N$.

A market outcome $(\mathbf{x}, \mathbf{p})$ is said to be a *market equilibrium* if:

(i) the market clears, i.e., all goods are fully allocated. Thus, for all j, $\sum_{i\in N} x_{ij} = 1$,

(ii) budget of all agents is exhausted, for all $i \in N$, $\sum_{j\in M} x_{ij} p_j = e_i$, and

(iii) agents only spend money on goods that give them maximum bang-per-buck, i.e., $(\mathbf{x}, \mathbf{p})$ is on MBB.

Given a market outcome $(\mathbf{x}, \mathbf{p})$ with $\mathbf{x}$ integral, we say it is *price envy-free up to one good* (pEF1) if for all $i, h \in N$ there is a good $j \in \mathbf{x}_h$ such that $\mathbf{p}(\mathbf{x}_i) \geq \mathbf{p}(\mathbf{x}_h \setminus \{j\})$. For integral market outcomes on MBB, the pEF1 condition implies the EF1 condition.

Lemma 1. *Let $(\mathbf{x}, \mathbf{p})$ be an integral market outcome on MBB. If $(\mathbf{x}, \mathbf{p})$ is pEF1 then $\mathbf{x}$ is EF1 and fPO.*

Proof. We first show that $(\mathbf{x}, \mathbf{p})$ forms a market equilibrium for the Fisher market instance (N, M, V, e), where for every $i \in N$, $e_i = \mathbf{p}(\mathbf{x}_i)$. It is easy to see that the market clears and the budget of every agent is exhausted. Further, $\mathbf{x}$ is on MBB as per our assumption. Now the fact that $\mathbf{x}$ is fPO follows from the First Welfare Theorem (Mas-Colell, Whinston, & Green, 1995), which shows that for any market equilibrium $(\mathbf{x}, \mathbf{p})$, the allocation $\mathbf{x}$ is fPO.

Since $(\mathbf{x}, \mathbf{p})$ is pEF1, for all pairs of agents $i, h \in N$, there is some good $j \in \mathbf{x}_h$ s.t. $\mathbf{p}(\mathbf{x}_i) \geq \mathbf{p}(\mathbf{x}_h \setminus \{j\})$. Since $(\mathbf{x}, \mathbf{p})$ is on MBB, $\mathbf{x}_i \subseteq \mathrm{MBB}_i$. Let α_i be the MBB-ratio of i at the prices $\mathbf{p}$. By definition of MBB, $v_i(\mathbf{x}_i) = \alpha_i \mathbf{p}(\mathbf{x}_i)$, and $v_i(\mathbf{x}_h \setminus \{j\}) \leq \alpha_i \mathbf{p}(\mathbf{x}_h \setminus \{j\})$. Combining these, we get that $\mathbf{x}$ is EF1. $\square$

Lemma 1 suggests that in order to obtain an EF1+PO allocation, it suffices to compute an integral market outcome $(\mathbf{x}, \mathbf{p})$ that is pEF1. In an outcome $(\mathbf{x}, \mathbf{p})$, we call an agent i with minimum $\mathbf{p}(\mathbf{x}_i)$ a *least spender* (LS) agent. If an outcome $(\mathbf{x}, \mathbf{p})$ is not pEF1, then there must be an agent h such that any LS i pEF1-envies h. That is, $\mathbf{p}(\mathbf{x}_h \setminus \{j\}) > \mathbf{p}(\mathbf{x}_i)$, for every $j \in \mathbf{x}_h$ and any LS i. We call such an agent h a *pEF1-violator* in $(\mathbf{x}, \mathbf{p})$.

Given a price vector $\mathbf{p}$, we define the MBB graph to be the bipartite graph $G = (N, M, E)$ where for an agent i and good j, $(i, j) \in E$ iff $j \in \mathrm{MBB}_i$. Such edges are called *MBB* edges. Given an accompanying allocation $\mathbf{x}$, we supplement G to include *allocation edges*, an edge between i and j if $j \in \mathbf{x}_i$.

For agents $i_0, \dots, i_\ell$ and goods $j_1, \dots, j_\ell$, a path $P = (i_0, j_1, i_1, j_2, \dots, j_\ell, i_\ell)$ in the MBB graph, where for all $1 \leq \ell' \leq \ell$, $j_{\ell'} \in \mathbf{x}_{i_{\ell'}} \cap \mathrm{MBB}_{i_{\ell'-1}}$, is called a *special* path. We define the *level* $\lambda(h; i_0)$ of an agent h w.r.t. i_0 to be half the length of the shortest special path from i_0 to h, and to be n if no such path exists. A path $P = (i_0, j_1, i_1, j_2, \dots, j_\ell, i_\ell)$ is an *alternating path* if it is special, and if $\lambda(i_0; i_0) < \lambda(i_1; i_0) < \cdots < \lambda(i_\ell; i_0)$, i.e., the path visits agents in increasing order of their level w.r.t. i_0. Further, the edges in an alternating path alternate between allocation and MBB edges. Typically, we consider alternating paths starting from a least spender (LS) agent. We use alternating paths to reduce the pEF1-envy of agent i towards agent h by transferring goods along the alternating path $(i_0, j_1, i_1, j_2, \dots, j_\ell, i_\ell)$, i.e., by transferring j_ℓ to $i_{\ell-1}$, etc. The structure of an alternating path ensures that transfers are done along MBB edges, implying that resulting allocations remain on MBB and hence preserve the fPO property of the allocation. Figure 1 provides an example of an alternating path from agent i_0 to agent i_3 involving agents i_1 and i_2 and goods j_1, j_2, and j_3.

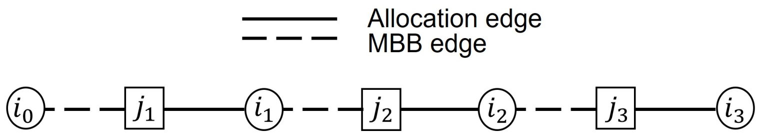

Figure 1: Example of an alternating path

Definition 1 (Component C_i of a least spender i)**.** For a least spender i, define C_i^ℓ as the set of all goods and agents that lie on alternating paths of length ℓ. Call $C_i = \bigcup_\ell C_i^\ell$ the *component* of i, the set of all goods and agents reachable from the least spender i through alternating paths.

We now illustrate the terms introduced in the above definitions through an example.

Example 1. Consider a fair division instance (N, M, V) with three agents $\{a_1, a_2, a_3\}$ and five goods $\{g_1, \dots, g_5\}$. The values are given by:

Consider the allocation $\mathbf{x}$ given by $\mathbf{x}_1 = \{g_1, g_2\}, \mathbf{x}_2 = \{g_3, g_4\}, \mathbf{x}_3 = \{g_5\}$ with associated price vector $\mathbf{p} = (6, 4, 2, 5, 2)$ defining the prices of the goods. The agents spending are given

	g_1	g_2	g_3	g_4	g_5
a_1	6	4	0	0	0
a_2	0	4	2	5	0
a_3	4	3	1	4	2

Table 2: A fair division instance

by $\mathbf{p}(\mathbf{x}_1) = 10$, $\mathbf{p}(\mathbf{x}_2) = 7$, and $\mathbf{p}(\mathbf{x}_3) = 2$. It can be checked that the MBB ratios of all agents are one and that the allocation is on MBB. Figure 2 describes the MBB graph associated with the allocation $(\mathbf{x}, \mathbf{p})$. In the graph, (a_2, g_2, a_1) is an alternating path. The least spender (LS) is a_3. The component of the LS is the agent a_3 and the good g_5.

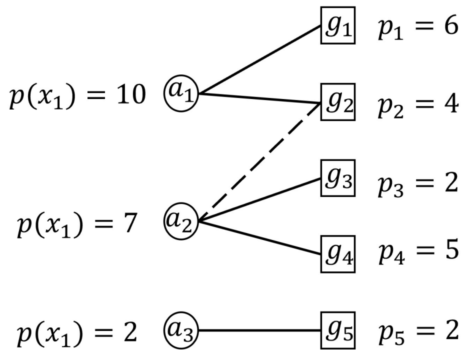

Figure 2: An allocation $(\mathbf{x}, \mathbf{p})$ and the associated MBB graph

3. Finding EF1+fPO Allocations of Goods

We now present the main algorithm of our paper. We show that given a fair division instance (N, M, V), Algorithm 1 returns an allocation $\mathbf{x}$ that is EF1 and fPO and terminates in time $\mathsf{poly}(n, m, U)$. Recall that U denotes the maximum number of distinct utility values an agent can achieve in any allocation.

Algorithm 1 starts with a welfare maximizing integral allocation $(\mathbf{x}, \mathbf{p})$, where $p_j = v_{ij}$ for $j \in \mathbf{x}_i$. If the allocation $\mathbf{x}$ is not EF1, then the outcome $(\mathbf{x}, \mathbf{p})$ cannot be pEF1. Thus, there must exist a pEF1-violator agent. Our algorithm aims to reduce pEF1-envy by transferring goods away from such pEF1-violator agents. Let L denote the set of least spenders. We first explore if a pEF1-violator is reachable via alternating paths starting from some least spender (LS), i.e., belongs to the component C_i of some LS i. If not, then we *raise the prices* of all goods in C_L, the union of components of all least spenders, until either (i) a new MBB edge gets added from an agent $h \in C_L$ to a good $j \notin C_L$ (corresponding to a price-rise of γ_1), or (ii) the spending of an agent $h \notin C_L$ becomes equal to the spending of the agents in L, i.e., a new agent becomes an LS (corresponding to a price-rise of γ_2).

Otherwise, we choose the minimum-index least spender i whose component C_i contains a pEF1-violator. We find an alternating path $P = (i, j_1, i_1, \ldots, j_\ell, i_\ell = h)$, from i to a pEF1-violator h, and it must be the case that $\mathbf{p}(\mathbf{x}_h \setminus \{j_\ell\}) > \mathbf{p}(\mathbf{x}_i)$. We then *transfer* the good j_ℓ from h to $i_{\ell-1}$. The choice of the minimum-index LS is to break ties among all LSs

Algorithm 1 Computing an EF1+fPO allocation of goods

Input: Fair division instance (N, M, V)
Output: An integral EF1+PO allocation $\mathbf{x}$

```
1: $(\mathbf{x}, \mathbf{p}) \leftarrow$ Initial welfare maximizing integral market allocation, where $p_j = v_{ij}$ for $j \in \mathbf{x}_i$.
2: while $(\mathbf{x}, \mathbf{p})$ is not pEF1 do
3:     $L \leftarrow \{k \in N : k \in \arg\min_{h \in N} \mathbf{p}(\mathbf{x}_h)\}$    ▷ set of LS
4:     if for all $i \in L$, there is no pEF1-violator in $C_i$ then    ▷ Perform price-rise step
5:         $\gamma_1 = \min_{h \in C_L \cap N, j \in M \setminus C_L} \frac{\alpha_h}{v_{hj}/p_j}$    ▷ Factor by which prices of goods in $C_L$ are
       raised until a new MBB edge appears from an agent in $C_L$ to a good outside $C_L$
6:         $\gamma_2 = \min_{i \in L, h \in N \setminus C_L} \frac{\mathbf{p}(\mathbf{x}_h)}{\mathbf{p}(\mathbf{x}_i)}$    ▷ Factor by which prices of goods in $C_L$ are raised
       until a new agent outside $C_L$ becomes a new LS
7:         $\beta = \min(\gamma_1, \gamma_2)$    ▷ Prise rise factor
8:         for $j \in C_L \cap M$ do    ▷ Raise prices of goods in $C_L$
9:             $p_j \leftarrow \beta p_j$
10:    else    ▷ Perform transfer step
11:        $i \leftarrow \min\{k \in L :$ there is a pEF1-violator in $C_k\}$    ▷ LS with smallest index
12:        Let $(i, j_1, i_1, \ldots, i_{\ell-1}, j_\ell, i_\ell)$ be an alternating path from $i$ to pEF1-violator $i_\ell$
13:        Transfer $j_\ell$ from $i_\ell$ to $i_{\ell-1}$
   return $\mathbf{x}$
```

whose component contains a pEF1-violator. This ensures that there are no back-to-back transfers, where good j transferred from agent h to h' via an alternating path starting from LS i and immediately in the next step from h' to h via an alternating path starting from another LS i', as our algorithm will only choose one LS $\min\{i, i'\}$. Moreover, this ensures that while the component C_i contains pEF1-violators for the minimum-index LS i, we only perform transfers from the pEF1-violators in C_i.

Our algorithm repeatedly performs either price-rise steps or transfer steps until the outcome $(\mathbf{x}, \mathbf{p})$ becomes pEF1. Before showing that the algorithm always terminates and analyzing its run-time, we illustrate its execution through an example.

Example 2. *(Algorithm execution)* We revisit the instance in Example 1, and begin with welfare maximizing allocation described in Example 1, i.e., $(\mathbf{x}, \mathbf{p})$ where $\mathbf{x}_1 = \{g_1, g_2\}, \mathbf{x}_2 = \{g_3, g_4\}, \mathbf{x}_3 = \{g_5\}$ and $\mathbf{p} = (6, 4, 2, 5, 2)$. Allocation $\mathbf{x}$ is not EF1 since $v_3(\mathbf{x}_3) = 2 < 3 = v_3(\mathbf{x}_1 \setminus g_1)$. Indeed, $(\mathbf{x}, \mathbf{p})$ is not pEF1 either, with a_3 pEF1-envying a_1, i.e., a_1 is a pEF1-violator. Since there is no alternating path from a_3 (the least spender) to a_1 (pEF1-violator), Algorithm 1 increases the price of g_5. A new MBB edge appears from a_3 to g_4 on a price-rise factor of $\gamma_1 = \frac{v_{35}/p_5}{v_{34}/p_4} = 1.25$, and a new LS appears on a price-rise factor of $\gamma_2 = \mathbf{p}(\mathbf{x}_2)/\mathbf{p}(\mathbf{x}_3) = 3.5$. Thus, the price of g_5 will be raised by $\gamma = \min(\gamma_1, \gamma_2) = 1.25$, i.e., the new price is $p_5 = 2.5$, and there an MBB edge from a_3 to g_4 is now present.

Since the allocation is not yet pEF1, Algorithm 1 checks if a pEF1-violator (a_3) belongs to the component of an LS (a_1). Since this is the case, Algorithm 1 identifies the shortest alternating path $(a_3, g_4, a_2, g_2, a_1)$ and makes the transfer of g_2 from a_1 to a_2. The new allocation is now $\mathbf{x}'$ given by $\mathbf{x}'_1 = \{g_1\}, \mathbf{x}'_2 = \{g_2, g_3, g_4\}, \mathbf{x}'_3 = \{g_5\}$. Since the allocation is still not pEF1 as a_3 pEF1-envies a_2, the algorithm transfers g_4 from the pEF1-violator

a_2 to a_3. This results in the allocation $\mathbf{x}''$ given by $\mathbf{x}''_1 = \{g_1\}, \mathbf{x}''_2 = \{g_2, g_3\}, \mathbf{x}''_3 = \{g_4, g_5\}$, which is pEF1, and thus Algorithm 1 terminates with an EF1+PO allocation.

We now proceed to analyze the run-time of the Algorithm 1. We will use the terms *time-step* or *iteration* interchangeably to denote either a transfer or a price-rise step. We say 'at time-step t' to refer to the state of the algorithm *just before* the event at t happens. We denote by $(\mathbf{x}^t, \mathbf{p}^t)$ the allocation and price vector at time-step t. First, we note that:

Lemma 2. *At any time-step t, $(\mathbf{x}^t, \mathbf{p}^t)$ is on MBB.*

Proof. We want to show that at every iteration of the algorithm, every agent owns goods from their MBB set. To see this, notice that this is the case in the algorithm's initial allocation in Line 1. Suppose we assume inductively that at iteration t, in the corresponding allocation $\mathbf{x}$, every agent buys MBB goods. We ensure that the goods are transferred along MBB edges, and thus, no transfer step causes the MBB condition of any agent to be violated. Consider a price-rise step, in which the prices of all goods in C_L, the component of the least spenders L, are increased by a factor of β. Since prices of these goods are raised, no agent $h \notin C_L$ prefers these goods over their own goods, and hence, they continue to own goods from their MBB set. For any agent $h \in C_L$ and a good $j \notin C_L$, the bang-per-buck that j gets from h before the price-rise is strictly less than her MBB ratio since j is not in the MBB set of h. Further, we never raise these prices beyond the creation of a new MBB edge from any agent $h \in C_L$ to some agent $j \in C_L$. Thus, the MBB condition is not violated for any agent $h \in C_L$.

In conclusion, the MBB condition is never violated for any agent throughout the algorithm, so the allocation is always on MBB. □

If Algorithm 1 terminates, then the final outcome $(\mathbf{x}, \mathbf{p})$ is pEF1. Since it is also on MBB, by Lemma 1, $\mathbf{x}$ is EF1+fPO. We now proceed towards the run-time analysis of Algorithm 1. First, we observe that prices are raised only until the spending of a new agent becomes equal to the spending of the least spenders.

Lemma 3. *The spending of the least spender(s) does not decrease as the algorithm progresses. Further, at any price-rise event t with price-rise factor β, the spending of the least spender(s) increases by a factor of β.*

Proof. The spending of a least spender can decrease if the least spender loses a good. This cannot happen since the only agents who lose goods are pEF1-violators, whose spending after removing one good exceeds that of the least spender. Moreover, if a good j is transferred from a pEF1-violator h directly to a LS i, then we have $\mathbf{p}(\mathbf{x}_h \setminus \{j\}) > \mathbf{p}(\mathbf{x}_i)$. Thus, even if h becomes a new LS after the transfer, the spending of the LS has only increased. Finally, note that in a price-rise step with price-rise factor β, the prices of all goods in C_L are increased by a factor of β. Thus, the spending of every agent in C_L, including the least spenders, increases by a factor of β. □

Next, we argue:

Lemma 4. *The number of iterations with the same set of least spenders is* $\mathsf{poly}(m, n)$.

Proof. Let us fix a set L of least spenders. We first argue that there can be at most n price-rise steps without any change in L. This is because of the following. Since L is unchanged, every price-rise step can only create a new MBB edge since a new LS will change L. Thus, each price-rise step causes a new agent to be added to C_L. Since there are at most n agents outside C_L at any given iteration, there can only be n price-rise steps.

We now argue that the number of transfer steps with the same set of LS between any two price steps is at most $\mathsf{poly}(m, n)$. Recall that our algorithm always uses the minimum-index least spender whose component contains a pEF1-violator. Because of this, all transfers are performed in a component C_i before moving to another component C_ℓ, for agents $i, \ell \in L$ with $i < \ell$. Thus, if a transfer from a pEF1-violator agent h to LS ℓ is performed, then h can not be in C_i at the time of transfer. (Barman et al., 2018a) used a potential function argument to show that the number of consecutive transfer steps with a fixed LS agent is at most $m \cdot n^2$. Thus, with our observation above, we can conclude that the number of consecutive transfers with the same set of LS is at most $m \cdot n^3 = \mathsf{poly}(m, n)$. □

The next lemma is key. We argue that between the time steps at which an agent i ceases to be an LS and subsequently becomes an LS again, her utility strictly increases.

Lemma 5. *Let t_0 be a time-step where agent i ceases to be an LS, and let t_ℓ be the first subsequent time step just after which i becomes the LS again. Then:*

$$v_i(\mathbf{x}_i^{t_\ell+1}) > v_i(\mathbf{x}_i^{t_0})$$

Note here that $v_i(\mathbf{x}_i^{t_0})$ is the utility of agent i just before *time-step t_0, and $v_i(\mathbf{x}_i^{t_\ell+1})$ her utility* just after *time-step t_ℓ.*

Proof. From Lemma 3, since i ceases to be an LS after time-step t_0, i must have received some good j at time step t_0. Since $j \in \mathrm{MBB}_i$ at t_0, $v_{ij} > 0$. Suppose i does not lose any good in any subsequent iterations until t_ℓ, then $\mathbf{x}_i^{t_\ell+1} \supseteq \mathbf{x}_i^{t_0} \cup \{j\}$, and hence $v_i(\mathbf{x}_i^{t_\ell+1}) \geq v_i(\mathbf{x}_i^{t_0} \cup \{j\}) = v_i(\mathbf{x}_i^{t_0}) + v_{ij} > v_i(\mathbf{x}_i^{t_0})$, using additivity of valuations.

On the other hand, suppose i does lose some goods between t_0 and t_ℓ. Let $t_k \in (t_0, t_\ell]$ be the last time-step when i loses a good, say j'. Let $t_1, \ldots, t_{k-1}$ be time-steps (in order) between t_0 and t_k when i experiences price-rise, and $t_{k+1}, \ldots, t_{\ell-1}$ be time-steps (in order) between t_k and t_ℓ when i experiences price-rise, until finally after the iteration t_ℓ agent i becomes the LS again. Let us define β_t to be the price-rise factor at the time-step t. If t is a price-rise step, $\beta_t > 1$, else we set $\beta_t = 1$. Hence $\beta_{t_1}, \ldots, \beta_{t_{k-1}}, \beta_{t_{k+1}}, \ldots, \beta_{t_{\ell-1}}$ are price-rise factors at the corresponding events $t_1, \ldots, t_{k-1}, t_{k+1}, \ldots, t_{\ell-1}$ and are all greater than 1. If t_ℓ is a price-rise event, let the price-rise factor be $\beta_{t_\ell} > 1$; and if not let $\beta_{t_\ell} = 1$. Note that t_k is not a price-rise event; hence, $\beta_{t_k} = 1$.

Using Lemma 3, together with the fact that i does not lose any good after t_k, we have:

$$\mathbf{p}^{t_\ell+1}(\mathbf{x}_i^{t_\ell+1}) \geq (\beta_{t_\ell}\beta_{t_{\ell-1}} \cdots \beta_{t_{k+1}})\mathbf{p}^{t_k}(\mathbf{x}_i^{t_k} \setminus \{j'\}). \tag{1}$$

The above may not be equality because in addition to experiencing price-rises during $t_{k+1}, \ldots, t_\ell$, agent i may also *gain* some new good. If i_k is a LS at t_k, then for agent i to lose the good j' it must be the case that:

$$\mathbf{p}^{t_k}(\mathbf{x}_i^{t_k} \setminus \{j'\}) > \mathbf{p}^{t_k}(\mathbf{x}_{i_k}^{t_k}). \tag{2}$$

Let i_t be a LS at time-step t. Then by repeatedly applying Lemma 3, we get:

$$\begin{aligned}\mathbf{p}^{t_k}(\mathbf{x}_{i_k}^{t_k}) &\geq \beta_{t_k-1} p^{t_k-1}(\mathbf{x}_{i_{t_k-1}}^{t_k-1}) \\ &\geq \cdots \geq (\beta_{t_k-1}\beta_{t_k-2}\cdots\beta_1)\mathbf{p}^{t_0}(\mathbf{x}_{i_{t_0}}^{t_0}) \\ &\geq (\beta_{t_{k-1}}\beta_{t_{k-2}}\cdots\beta_{t_1})\mathbf{p}^{t_0}(\mathbf{x}_i^{t_0}) \ , \end{aligned} \tag{3}$$

where the last transition follows from the facts that (i) each $\beta_t \geq 1$, (ii) $\{t_1, \ldots, t_{k-1}\} \subseteq \{1, \ldots, t_k - 1\}$, and (iii) $i_{t_0} = i$ since i is a least spender at t_0. Putting (1), (2) and (3) together, we get:

$$\mathbf{p}^{t_\ell+1}(\mathbf{x}_i^{t_\ell+1}) > (\Pi_{r=1}^{\ell}\beta_{t_r})\mathbf{p}^{t_0}(\mathbf{x}_i^{t_0}) \ . \tag{4}$$

Let α_i^t denote the MBB-ratio of i at the time step t. Observe that in every price-rise event with price-rise factor β, the MBB ratio of any agent experiencing the price-rise decreases by β. Further, the MBB ratio of any agent does not change unless she experiences a price-rise step. Thus:

$$\alpha_i^{t_\ell+1} = \frac{\alpha_i^{t_0}}{(\beta_{t_\ell}\beta_{t_{\ell-1}}\cdots\beta_{t_{k+1}})(\beta_{t_{k-1}}\beta_{t_{k-2}}\cdots\beta_{t_1})} \ . \tag{5}$$

Therefore, using the fact that the allocation is on MBB edges, and with (4) and (5), we have:

$$\begin{aligned} v_i(\mathbf{x}_i^{t_\ell+1}) &= \alpha_i^{t_\ell+1}\mathbf{p}^{t_\ell+1}(\mathbf{x}_i^{t_\ell+1}) && (\mathbf{x}^{t_\ell+1} \text{ is on MBB}) \\ &> \frac{\alpha_i^{t_0}}{(\Pi_{r=1}^{\ell}\beta_{t_r})}(\Pi_{r=1}^{\ell}\beta_{t_r})\mathbf{p}^{t_0}(\mathbf{x}_i^{t_0}) && (\text{From (4) and (5)}) \\ &= \alpha_i^{t_0}\mathbf{p}^{t_0}(\mathbf{x}_i^{t_0}) = v_i(\mathbf{x}_i^{t_0}), && (\mathbf{x}^{t_0} \text{ is on MBB}) \end{aligned}$$

as claimed. □

Using the above lemmas, we show:

Lemma 6. *Algorithm 1 terminates in time* $\mathsf{poly}(n, m, U)$.

Proof. Consider any agent i. From Lemma 5, it is clear that every time i becomes the LS again her utility has strictly increased compared to her utility the last time she was an LS. The number of utility values that i can have is U_i; hence, we conclude that the number of times she stops being an LS and becomes LS again is at most U_i. Since there are n agents, and each agent i can become the LS again at most U_i times, we have that after $\mathsf{poly}(n, \max_{i\in N} U_i)$ changes in the set of least spenders, there will be no changes further in the set of least spenders. After this, in at most n more price-rise steps, either the allocation becomes pEF1, or all agents get added to C_L since no new agent becomes an LS on raising prices. Further, the number of transfers with the same set of least spenders is at most $\mathsf{poly}(m, n)$ (Lemma 4). This shows that Algorithm 1 terminates in time $\mathsf{poly}(n, m, U)$. □

Putting it all together, we conclude:

Theorem 3. *Let $I = (N, M, V)$ be a fair division instance. Then, an allocation that is both EF1 and fPO can be computed in time* $\mathsf{poly}(n, m, U)$.

Observe that in any allocation and for any agent, the minimum utility is 0, and the maximum utility is mv_{max}, where $v_{max} = \max_{i,j} v_{ij}$. Since the utility values are integral, we have $U \leq mv_{max} + 1$. Thus, Algorithm 1 computes an EF1+fPO allocation in pseudo-polynomial time.

Theorem 4. *Given a fair division instance $I = (N, M, V)$, an allocation that is both EF1 and fPO can be computed in time $\mathsf{poly}(n, m, v_{max})$, where $v_{max} = \max_{i,j} v_{ij}$. In particular, when $v_{max} \leq \mathsf{poly}(m, n)$, an EF1+fPO allocation can be computed in $\mathsf{poly}(m, n)$ time.*

The guarantee of EF1+fPO offered by our algorithm is stronger than the guarantee of EF1+PO provided by the algorithm of Barman et al. (2018a). We next turn our attention to k-ary instances where k is a constant. First, we observe that for such instances, the maximum number of different utility values any agent can get is at most $\mathsf{poly}(m)$.

Lemma 7. *In a k-ary fair division instance (N, M, V) with constant k, $U \leq \mathsf{poly}(m)$.*

Proof. For any agent i, let $\{v_i^\ell\}_{\ell \in [k]}$ be the different utility values i has for the goods. In an allocation $\mathbf{x}$, let $m_i^\ell \in \mathbb{Z}_{\geq 0}$ be the number of goods in $\mathbf{x}_i$ with value v_i^ℓ. Then, agent i's utility is simply: $v_i(\mathbf{x}_i) = m_i^1 v_i^1 + \cdots + m_i^k v_i^k$. Since each $0 \leq m_i^\ell \leq m$, the number of possible utility values that i can get in any allocation is at most $(m+1)^k$, which is $\mathsf{poly}(m)$ since k is constant. Thus $U \leq \mathsf{poly}(m)$. □

Therefore, using Lemma 7, Theorem 3 gives:

Theorem 5. *Given a k-ary fair division instance $I = (N, M, V)$ where k is a constant, an allocation that is both EF1 and fPO can be computed in time $\mathsf{poly}(m, n)$.*

Remark 6. *We note that our techniques can also be used to show that an allocation that is weighted-EF1 (wEF1) and fPO exists and can be computed in pseudo-polynomial time. Here, an allocation $\mathbf{x}$ is wEF1 if for all agents i, h we have $\frac{v_i(\mathbf{x}_i)}{w_i} \geq \frac{v_i(\mathbf{x}_h \setminus j)}{w_h}$ for some $j \in \mathbf{x}_h$, where w_k denotes the weight of agent k. We can modify Algorithm 1 for the weighted setting by considering weighted-LS instead of LS, as the agents with minimum $\frac{\mathbf{p}(\mathbf{x}_i)}{w_i}$, and performing transfer from agent h if it violates weighted-pEF1, i.e., if $\frac{\mathbf{p}(\mathbf{x}_h \setminus j)}{w_h} > \frac{\mathbf{p}(\mathbf{x}_i)}{w_i}$ for every $j \in \mathbf{x}_h$ and a weighted-LS i. Lemmas 2 and 4 still hold as before. Lemma 3 holds with weighted-spending instead of spending. With these modifications, the key Lemma 5 can be shown as before, thus implying pseudo-polynomial run time for computing an EF1+fPO allocation in the weighted case as well.*

4. Finding EQ1+fPO Allocations of Goods

We now show that Algorithm 2 finds an EQ1+fPO allocation given a fair division instance with positive values. We require the values to be positive because instances with zero values might not even admit an allocation that is EQ1+PO (Freeman et al., 2019). Algorithm 2 is similar to Algorithm 1, except that it works with values instead of spendings of agents since we desire EQ1 allocations instead of EF1.

Algorithm 2 starts with a welfare maximizing integral allocation $(\mathbf{x}, \mathbf{p})$, where $p_j = v_{ij}$ for $j \in \mathbf{x}_i$. We refer to an agent with the least utility as an LU agent and let L be the

Algorithm 2 Computing an EQ1+fPO allocation of goods

Input: Positive-valued fair division instance (N, M, V)

Output: An integral EQ1+PO allocation $\mathbf{x}$

```
1: $(\mathbf{x}, \mathbf{p}) \leftarrow$ Initial welfare maximizing integral market allocation, where $p_j = v_{ij}$ for $j \in \mathbf{x}_i$.
2: while $\mathbf{x}$ is not EQ1 do
3:     $L \leftarrow \{i \in N : i \in \mathsf{argmin}_{h \in N} v_h(\mathbf{x}_h)\}$                ▷ set of LU agents
4:     if for all $i \in L$, there is no EQ1-violator in $C_i$ then          ▷ Perform price-rise step
5:         $\beta = \min_{h \in C_L \cap N, j \in M \setminus C_L} \frac{\alpha_h}{v_{hj}/p_j}$          ▷ Factor by which prices of goods in $C_L$ are
   raised until a new MBB edge appears from an agent in $C_L$ to a good outside $C_L$
6:         for $j \in C_L \cap M$ do
7:             $p_j \leftarrow \beta p_j$
8:     else                                                        ▷ Perform transfer step
9:         $i \leftarrow \min\{k \in L :$ there is an EQ1-violator in $C_k\}$          ▷ LU with smallest index
10:        Let $(i, j_1, i_1, \dots, i_{\ell-1}, j_\ell, i_\ell)$ be an alternating path from $i$ to EQ1-violator $i_\ell$
11:        Transfer $j_\ell$ from $i_\ell$ to $i_{\ell-1}$
   return $\mathbf{x}$
```

set of LU agents. If the allocation $\mathbf{x}$ is not EQ1, then exist an EQ1-violator agent h, i.e., $v_h(\mathbf{x}_h \setminus \{j\}) > v_i(\mathbf{x}_i)$, for any $j \in \mathbf{x}_h$ and any LU agent i. Similar to Algorithm 1, we first explore if such an EQ1-violator belongs to the component C_i of some LU agent i. If not, *we raise the prices* of all goods in C_L until a new MBB edge gets added from an agent $h \in C_L$ to a good $j \notin C_L$.

Otherwise, we choose a minimum-index LU agent i whose component C_i contains an EQ1-violator. We find an alternating path $P = (i, j_1, i_1, \dots, j_\ell, i_\ell = h)$, from i to a EQ1-violator h, and it must be the case that $v(\mathbf{x}_h \setminus \{j_\ell\}) > v(\mathbf{x}_i)$. We then *transfer* the good j_ℓ from h to $i_{\ell-1}$.

Thus, the algorithm performs price-rise or transfer steps while the allocation is not EQ1. If the algorithm terminates, the allocation must be EQ1 (Line 2). By arguments similar to Lemma 2, we can show that the outcome throughout the execution of the algorithm is always on MBB and hence is fPO.

We now prove that the algorithm terminates. Similar to Lemma 3, we can argue that the utility of the LU agents(s) does not decrease in the algorithm's run. Further, similar to Lemma 5, we can show the following key lemma:

Lemma 8. *Let t_0 be a time-step where agent i ceases to be an LU agent, and let t_ℓ be the first subsequent time step just after which i becomes the LU agent again. Then:*

$$v_i(\mathbf{x}_i^{t_\ell+1}) > v_i(\mathbf{x}_i^{t_0}).$$

Proof. From Lemma 3, i must have received some good j at time step t_0. Since $j \in \mathrm{MBB}_i$ at t_0, $v_{ij} > 0$. Suppose i does not lose any good in any subsequent iterations, then $\mathbf{x}_i^{t_\ell+1} \supseteq \mathbf{x}_i^{t_0} \cup \{j\}$, and hence $v_i(\mathbf{x}_i^{t_\ell+1}) > v_i(\mathbf{x}_i^{t_0})$. On the other hand, suppose i does lose some goods. Let $t_k \in (t_0, t_\ell]$ be a subsequent time step where i loses a good j' for the last time. Since i does not lose any good after t_k, we have:

$$v_i(\mathbf{x}_i^{t_\ell+1}) \geq v_i(\mathbf{x}_i^{t_k} \setminus \{j'\}). \tag{6}$$

If i_k is a LU at t_k, then for i to lose j' it must be that:

$$v_i(\mathbf{x}_i^{t_k} \setminus \{j'\}) > v_{i_{t_k}}(\mathbf{x}_{i_k}^{t_k}). \tag{7}$$

Finally, since the utility of an LU agent does not decrease:

$$v_{i_{t_k}}(\mathbf{x}_{i_k}^{t_k}) \geq v_i(\mathbf{x}_i^{t_0}). \tag{8}$$

Putting (6), (7) and (8) together, we get:

$$v_i(\mathbf{x}_i^{t_\ell+1}) > v_i(\mathbf{x}_i^{t_0})$$

as required. □

Using the above, as argued in Lemma 6, we can show:

Lemma 9. *Algorithm 2 terminates in time* $\mathsf{poly}(n, m, U)$.

Proof. Consider any agent i. From Lemma 8, it is clear that every time an agent i becomes a LU agent again, her utility has strictly increased compared to her utility the last time she was a LU agent. The number of utility values that i can have is U_i, and hence we conclude that the number of times she stops being an LU agent and becomes an LU agent again is at most U_i. Since there are n agents, and each agent i can become the LU agent again at most U_i times, we have that after $\mathsf{poly}(n, \max_{i \in N} U_i)$ changes in the set of LU agents, there will be no changes further in the set of LU agents. Further, using an analysis similar to Lemma 4, we can show that the number of transfers with the same set of LU agents is at most $\mathsf{poly}(m, n)$. This shows that Algorithm 1 terminates in time $\mathsf{poly}(n, m, U)$. □

We conclude:

Theorem 7. *Let* $I = (N, M, V)$ *be a positive-valued fair division instance. Then, an allocation that is both EQ1 and fPO can be computed in time* $\mathsf{poly}(n, m, U)$.

As argued before, we have $U \leq mv_{max} + 1$. This gives:

Theorem 8. *Given a fair division instance* $I = (N, M, V)$*, an allocation that is EQ1 and fPO can be computed in time* $\mathsf{poly}(n, m, v_{max})$*, where* $v_{max} = \max_{i,j} v_{ij}$*. In particular, when* $v_{max} \leq \mathsf{poly}(m, n)$*, an EQ1+fPO allocation can be computed in* $\mathsf{poly}(m, n)$ *time.*

Finally using Lemma 7, Theorem 7 becomes:

Theorem 9. *Given a k-ary fair division instance* $I = (N, M, V)$ *where k is a constant, an allocation that is EQ1 and fPO can be computed in time* $\mathsf{poly}(m, n)$.

Remark 10. We remark that our techniques can be used to show that EQ1+fPO allocations of *chores* can be computed in pseudo-polynomial time and in polynomial-time for k-ary instances with constant k. In the chores model, agent i incur a cost or disutility $c_{ij} \geq 0$ on being assigned the chore j. For chores, an allocation $\mathbf{x}$ is said to be EQ1 if for all agents i, h, there exists a chore $j \in \mathbf{x}_i$ s.t. $c_i(\mathbf{x}_i \setminus \{j\}) \leq c_h(\mathbf{x}_h)$. Note that while comparing bundles, an agent removes a chore from her own bundle in the case of chores, while an agent removes a good from other's bundle in the case of goods. To obtain an EQ1+fPO allocation of chores, Algorithm 2 can be modified to perform transfers from an EQ1-violator agent h, where $c_h(\mathbf{x}_h \setminus j) > c_i(\mathbf{x}_i)$ for every $j \in \mathbf{x}_h$ and a least disutility agent i. The convergence analysis for this modification of the algorithm for chores is similar to that of Algorithm 2. We also note that in general, the existence of EF1+PO allocations of chores is open.

5. Finding EF1+PO Allocations for Constant n

In this section, we show how to compute an EF1+PO allocation in polynomial time when n, the number of agents, is a constant. Our algorithm relies on the fact that there *exists* an EF1+fPO allocation for every instance and thus aims to find such an allocation by effectively enumerating over all fPO allocations. For constant n, we argue that this enumeration takes time $\mathsf{poly}(m)$. We borrow some terminology from (Brânzei & Sandomirskiy, 2019) to describe this enumeration technique.

Given a fractional allocation $\mathbf{z}$, the *consumption graph* $G_{\mathbf{z}}$ is defined to be a bipartite graph (N, M, E), where $(i, j) \in E$ iff $z_{ij} > 0$. Let $P = (i_1, j_1, i_2, j_2, \dots, i_K, j_K, i_{K+1})$, be a path in a consumption graph (N, M), where $K \geq 1$, and $v_{ij} > 0$ for any $(i, j) \in P$. We define the product of utilities along P as:

$$\pi(P) = \prod_{k=1}^{K} \frac{v_{i_k j_k}}{v_{i_{k+1} j_k}}. \tag{9}$$

When $i_{K+1} = i_1$, P is a cycle. We now characterize fPO allocations based on the properties of their associated consumption graphs.

Lemma 10 (Brânzei and Sandomirskiy (2019))**.** *An allocation* $\mathbf{z}$ *is fPO iff for every cycle* C *in its consumption graph,* $\pi(C) = 1$*.*

Proof. (sketch) See Corollary 16 of (Brânzei & Sandomirskiy, 2019) for the full proof. Intuitively, if $\pi(C) > 1$, then transferring small amounts of goods along the cycle will result in an allocation $\mathbf{z}'$, which is a Pareto-improvement over $\mathbf{z}$, contradicting the fact that $\mathbf{z}$ is fPO. Likewise, if $\pi(C) < 1$, performing the transfer in the opposite order results in a Pareto improvement. □

Thus, the existence of cycles C in a consumption graph of an allocation with $\pi(C)$ depends on algebraic properties of the (non-zero) values v_{ij} of the instance. This motivates the definition of *non-degenerate instance* as an instance where the values v_{ij} share no multiplicative relationship. Formally:

Definition 2 (Non-degenerate instance)**.** A fair division instance $I = (N, M, V)$ is said to be non-degenerate if for every path $P = (i_1, j_1, i_2, j_2, \dots, i_K, j_K, i_{K+1})$, in the complete bipartite graph (N, M), where $K \geq 1$, and $v_{ij} > 0$ for any $(i, j) \in P$, it holds that $\pi(P) \neq 1$.

For an allocation $\mathbf{z}$, let $\mathbf{u}(\mathbf{z}) \in \mathbb{R}^n$ be the corresponding utility vector, where $\mathbf{u}(\mathbf{z})_i = v_i(\mathbf{z}_i)$ for each $i \in N$. Brânzei and Sandomirskiy (2019) showed that for each fPO utility vector $\mathbf{u}$ of a non-degenerate instance, there is a unique feasible (fractional) allocation $\mathbf{z}$ such that $\mathbf{u}(\mathbf{z}) = \mathbf{u}$. Brânzei and Sandomirskiy (2019) showed how to enumerate fPO allocations using the following definition.

Definition 3 (Rich family of graphs)**.** A collection of bipartite graphs $\mathcal{G}$ is said to be *rich* for a given instance (N, M, V) if for any fPO utility vector $\mathbf{u}$, there is a feasible allocation $\mathbf{z}$ with $\mathbf{u}(\mathbf{z}) = \mathbf{u}$ such that the consumption graph $G_{\mathbf{z}}$ is in the collection $\mathcal{G}$.

Thus, a rich family of graphs contains the consumption graphs of every fPO utility vector for the instance. Brânzei and Sandomirskiy (2019) show how to construct a rich family of graphs $\mathcal{G}$ for every instance.

Theorem 11 (Proposition 23 of (Brânzei & Sandomirskiy, 2019))**.** *For constant number of agents n, a rich family of graphs $\mathcal{G}$ can be constructed in time $O(m^{\frac{n(n-1)}{2}+1})$ and has at most $(2m+1)^{\frac{n(n-1)}{2}+1}$ elements.*

Our algorithm for finding an EF1+fPO allocation in a non-degenerate instance I with a constant number of agents is as follows. We generate a rich family of graphs in $\mathsf{poly}(m)$ time since n is constant. We know there exists an integral EF1+fPO allocation $\mathbf{x}$, for instance I, with corresponding utility profile $\mathbf{u}(\mathbf{x})$. Since I is non-degenerate, $\mathbf{x}$ is the only allocation corresponding to the utility profile $\mathbf{u}(\mathbf{x})$. Now since the collection $\mathcal{G}$ is rich, for the utility profile $\mathbf{u}(\mathbf{x})$, the consumption graph $G_\mathbf{x}$ of $\mathbf{x}$ is present in $\mathcal{G}$. Since $\mathbf{x}$ is integral, the degree of every good in $G_\mathbf{x}$ is one. Thus, we can find $\mathbf{x}$ by iterating over every graph G in $\mathcal{G}$, checking if the degree of every good in G is one, i.e., corresponds to an integral allocation, and then checking if the integral allocation is EF1. This algorithm runs in polynomial time since $\mathcal{G}$ has $\mathsf{poly}(m)$ graphs as n is constant, and checking if an allocation is EF1 can also be done in $\mathsf{poly}(m)$ time.

We now show how to adapt our algorithm for all instances, not just non-degenerate instances. Given a fair division instance $I = (N, M, V)$, we use the perturbation technique of Duan, Garg, and Mehlhorn (2016) to construct a non-degenerate instance $I' = (N, M, V')$ as follows. For each $i \in N$ and $j \in M$, we choose a distinct prime number q_{ij}. The prime number theorem implies that for every integer Q, there are at least $\frac{Q}{2\ln Q}$ primes less than or equal to Q. Thus for $Q = 8nm\ln(nm)$, there are at least nm primes at most Q, which can be computed in $O(Q \ln Q)$ time. The values in the perturbed instance I' are then defined as $v'_{ij} = v_{ij} \cdot q_{ij}^{\varepsilon}$, for a small constant ε given by $\varepsilon := \log_Q(1 + \frac{1}{2mv_{max}}) \in (0, 1)$.

We now run our algorithm on the non-degenerate instance I' and compute an EF1+fPO allocation $\mathbf{x}$. We argue that $\mathbf{x}$ is EF1+PO for the original instance I as well.

Lemma 11. *If $\mathbf{x}$ is EF1 for I', then $\mathbf{x}$ is EF1 for I.*

Proof. Consider any agent $i \in N$ and two sets $S, T \subseteq M$ such that $v_i(S) > v_i(T)$. First note:

$$v'_i(S) - v_i(S) = \sum_{j \in S} v_{ij} \cdot (q_{ij}^{\varepsilon} - 1) \geq 0.$$

Next, observe that:

$$v'_i(T) - v_i(T) = \sum_{j \in T} v_{ij} \cdot (q_{ij}^{\varepsilon} - 1) \leq m \cdot v_{max} \cdot (Q^{\varepsilon} - 1) = \frac{1}{2},$$

where the last equality used the definition of ε. Putting the above two inequalities together, we have:

$$v'_i(S) - v'_i(T) \geq v_i(S) - v_i(T) - \frac{1}{2} > 0,$$

where the last inequality holds because $v_i(S) - v_i(T) \geq 1$, since the original valuations are integral.

Suppose that $\mathbf{x}$ is EF1 for the instance I' and not EF1 for the instance I. Then there are two agents i, h s.t. for all $j \in \mathbf{x}_h$: $v_i(\mathbf{x}_h \setminus \{j\}) > v_i(\mathbf{x}_i)$. Then the argument above implies that $v'_i(\mathbf{x}_h \setminus \{j\}) > v'_i(\mathbf{x}_i)$ for every $j \in \mathbf{x}_h$. This contradicts our assumption that $\mathbf{x}$ is EF1 for I'. □

Lemma 12. *If* $\mathbf{x}$ *is fPO for* I'*, then* $\mathbf{x}$ *is PO for* I*.*

Proof. The Second Welfare Theorem (Mas-Colell et al., 1995) states that for any fPO allocation $\mathbf{x}$ of an instance $([n], [m], V)$, there exists a set of non-negative budgets $\{e_i\}_{i\in[n]}$ and a set of prices of the good $\mathbf{p} \geq 0$, such that $(\mathbf{x}, \mathbf{p})$ is a market equilibrium for the Fisher market instance $([n], [m], V, \{e_i\}_{i\in[n]})$.

Therefore, since $\mathbf{x}$ is fPO for I', by the Second Welfare Theorem, there exists a set of prices $\mathbf{p}$ s.t. $(\mathbf{x}, \mathbf{p})$ is a market equilibrium for an associated Fisher market instance. In particular, $(\mathbf{x}, \mathbf{p})$ is on MBB. For an agent i, let $\alpha'_i = \max_{j\in M} v'_{ij}/p_j$, and let $\alpha_i = \max_{j\in M} v_{ij}/p_j$. Since the prices can be scaled, we can assume that for all $j \in M$, $1 \leq p_j \leq 2$.

For any agent i and good j, by the definition of v'_{ij} and $\varepsilon := \log_Q(1 + \frac{1}{2mv_{max}})$, we have that:

$$v_{ij} \leq v'_{ij} \leq v_{ij} \cdot (1 + \delta),$$

where $\delta = \frac{1}{2mv_{max}}$. This means that $\alpha'_i \geq \alpha_i$ for every $i \in N$. Further, since $(\mathbf{x}, \mathbf{p})$ is on MBB for I', for any $i \in N$:

$$\mathbf{p}(\mathbf{x}_i) = \frac{v'_i(\mathbf{x}_i)}{\alpha'_i} \leq \frac{(1+\delta)v_i(\mathbf{x}_i)}{\alpha_i},$$

which gives:

$$\sum_{i\in N} \frac{v_i(\mathbf{x}_i)}{\alpha_i} \geq \sum_{i\in N} \frac{\mathbf{p}(\mathbf{x}_i)}{1+\delta} \geq \frac{\mathbf{p}(M)}{1+\delta} \tag{10}$$

Suppose $\mathbf{x}$ is not PO for the instance I. Then there is some allocation $\mathbf{y}$ that Pareto-dominates $\mathbf{x}$, i.e., for every $i \in N$, $v_i(\mathbf{y}_i) \geq v_i(\mathbf{x}_i)$, and for some $h \in N$, $v_h(\mathbf{y}_h) \geq v_h(\mathbf{x}_h) + 1$, since the valuations v_{ij} are integral. Further since α_i is the maximum bang-per-buck ratio for an agent i at prices $\mathbf{p}$, we have for every $i \neq h$: $\mathbf{p}(\mathbf{y}_i) \geq \frac{v_i(\mathbf{y}_i)}{\alpha_i} \geq \frac{v_i(\mathbf{x}_i)}{\alpha_i}$ and $\mathbf{p}(\mathbf{y}_h) \geq \frac{v_h(\mathbf{y}_h)}{\alpha_h} \geq \frac{v_h(\mathbf{x}_h)+1}{\alpha_h}$. This gives:

$$\begin{aligned}\sum_{i\in N} \frac{v_i(\mathbf{x}_i)}{\alpha_i} &= \sum_{i\in N\setminus\{h\}} \frac{v_i(\mathbf{x}_i)}{\alpha_i} + \frac{v_h(\mathbf{x}_h)}{\alpha_h} \\ &\leq \sum_{i\in N\setminus\{h\}} \mathbf{p}(\mathbf{y}_i) + \mathbf{p}(\mathbf{y}_h) - \frac{1}{\alpha_h} \\ &= \mathbf{p}(M) - 1/\alpha_h.\end{aligned} \tag{11}$$

Putting (10) and (11) together, we get:

$$\mathbf{p}(M) - 1/\alpha_h \geq \frac{\mathbf{p}(M)}{1+\delta},$$

which simplifies to:

$$\delta\alpha_h\mathbf{p}(M) \geq 1 + \delta.$$

Notice however that $\alpha_h \leq v_{max}$, and $\mathbf{p}(M) \leq 2m$, and hence:

$$\delta\alpha_h\mathbf{p}(M) \leq \delta \cdot v_{max} \cdot 2m = 1,$$

which is a contradiction. Hence $\mathbf{x}$ must be PO. □

Thus, we have shown:

Theorem 12. *Given a fair division instance $I = (N, M, V)$, where n is constant, an EF1+PO allocation can be computed in* $\mathsf{poly}(m)$ *time.*

Finally, we note that the techniques in Lemma 11 also extend easily to EQ1. Since we showed that an EQ1+fPO allocation is guaranteed to exist for positive instances:

Theorem 13. *Given a positive fair division instance $I = (N, M, V)$, where n is constant, an EQ1+PO allocation can be computed in* $\mathsf{poly}(m)$ *time.*

6. k-ary Instances with Constant n and k

We now consider k-ary fair division instances (N, M, V) where both k and n (number of agents) are constant.

Let $\mathcal{X}$ be the set of all allocations for the instance I. For each agent $i \in N$, let $T_i = \{v_i(\mathbf{x}_i) : \mathbf{x} \in \mathcal{X}\}$, the set of different utility values i can get from any allocation. Let $U = \max_{i \in N} |T_i|$. From Lemma 7, we know U is at most $\mathsf{poly}(m)$. Define $T = T_1 \times \cdots \times T_n$. We note that $|T| \leq (\mathsf{poly}(m))^n = \mathsf{poly}(m)$, since n is constant, and can be computed in $\mathsf{poly}(m)$-time.

To solve certain fair division problems for such instances, we enumerate over each entry $(u_1, \ldots, u_n)$ of T and check if there is a feasible allocation $\mathbf{x}$ in which each agent i gets utility exactly u_i. The next lemma shows that the latter can be done efficiently.

Lemma 13. *Given a vector $(u_1, \ldots, u_n) \in T$, it can be checked in* $\mathsf{poly}(m)$*-time whether there is a feasible allocation $\mathbf{x}$ s.t. for all agents i, $v_i(\mathbf{x}_i) = u_i$.*

Proof. For each agent $i \in N$, let $S_i = \{v_{ij} : j \in M\}$ be the set of values i has for the goods. For each good $j \in M$, define $\mathsf{label}(j)$ to be the position of the vector $(v_{1j}, \ldots, v_{nj})$ when the elements of $S_1 \times \cdots \times S_n$ are ordered lexicographically. Let L be the set of labels. Clearly $|L| \leq |S_1| \cdots |S_n| = O(1)$ since each $|S_i| \leq k$, and k and n are constants. Essentially, this means that there are $|L| = O(1)$ different *types* of goods. For a label $1 \leq \ell \leq |L|$, and for any agent i, let $v_{i\ell}$ equal v_{ij} for any good j with $\mathsf{label}(j) = \ell$. Further, let m_ℓ be the number of goods with label ℓ. All goods can be labeled in $\mathsf{poly}(m)$ time.

For each agent $i \in N$, let $T_i = \{v_i(\mathbf{x}_i) : \mathbf{x} \in \mathcal{X}\}$, the set of different utility values i can get from any allocation. Let $U = \max_{i \in N} |T_i|$. From Lemma 7, we know U is at most $\mathsf{poly}(m)$. Define $T = T_1 \times \cdots \times T_n$. We note that $|T| \leq (\mathsf{poly}(m))^n = \mathsf{poly}(m)$, since n is constant.

To solve certain fair division problems for such instances, we simply enumerate over each entry $(u_1, \ldots, u_n)$ of the set T and check if there is a feasible allocation $\mathbf{x}$ in which each agent i gets utility exactly u_i.

For the latter, we define integer variables $m_{i\ell}$ for each $i \in N$ and $1 \leq \ell \leq |L|$, which represents the number of goods with label ℓ assigned to agent i. Now consider the following linear system:

$$\forall i \in N : \quad \sum_{\ell=1}^{|L|} m_{i\ell} v_{i\ell} = u_i$$

$$\forall \ell \in [|L|]: \quad \sum_{i \in N} m_{i\ell} = m_\ell$$
$$\forall i \in N, \forall \ell \in [|L|]: \quad 0 \le m_{i\ell} \le m_\ell$$

This system has $n|L|$ variables, and each variable $m_{i\ell}$ can take at most $m+1$ values. Therefore, this system has a constant dimension, and each variable is in a bounded range. Thus, by simple enumeration, we can check in $\mathsf{poly}(m)$ time whether this system has a solution or not. If it does, then an allocation $\mathbf{x}$ exists, which gives every agent i a utility of u_i.

Therefore, given a vector $(u_1, \dots, u_n) \in T$, checking if there is a feasible allocation $\mathbf{x}$ in which each agent i gets utility exactly u_i can be done in $\mathsf{poly}(m)$-time. □

By iterating through T, we can prepare a list of feasible utility vectors (and corresponding allocations) that satisfy our fairness and efficiency criteria in $\mathsf{poly}(m)$-time.

Theorem 14. *For k-ary instance $I = (N, M, V)$ where both k and the number of agents n are constants, we can compute in $\mathsf{poly}(m)$ time (i) an MNW allocation, (ii) a leximin optimal allocation, (iii) a $\mathcal{F}$+fPO allocation (when it exists) where $\mathcal{F}$ is any polynomial-time checkable property.*

7. PLS Membership of Finding an EF1+fPO Allocation

In this section, we show that the problem of computing an EF1+fPO allocation lies in the complexity class PLS (Johnson, Papadimitriou, & Yannakakis, 1988). Essentially, PLS captures local search problems whose local optimality can be verified in polynomial time. We, therefore, phrase the problem of computing an EF1+fPO allocation in a fair division instance $I = (N, M, V)$ as a local search problem Φ. We closely follow our Algorithm 1, which computes an EF1+fPO allocation and shows that it has the structure of a local search problem. We now describe the solution space, cost function, and neighborhood structure of Φ, following which we show that the local maxima of Φ correspond to EF1+fPO allocations for the instance I.

Solution space. We first define a configuration space as follows. Let $\mathbf{x}^0$ be a specific initial integral fPO allocation. Each element of the configuration space is of the form $(\mathbf{x}, L, \varphi)$, where $\mathbf{x}$ is an integral fPO allocation, $L \subseteq N$ is a set of agents, and $\varphi \in \{0, 1, \dots, U\}^n$, where U is the maximum utility an agent can get in any allocation. Clearly, the solution space is finite and can be represented in size polynomial in the representation of the instance I. Moreover, since it can be checked in polynomial time via a linear program whether a given allocation is fPO (Barman et al., 2018a), we can efficiently check if a given tuple $(\mathbf{x}, L, \varphi)$ is a member of the configuration space or not.

Given a configuration $(\mathbf{x}, L, \varphi)$, we use the convex program (12) below to find a set of unique prices $\mathbf{p}$ s.t. $\mathbf{p}$ s.t. $\sum_j p_j = n$, $(\mathbf{x}, \mathbf{p})$ is on MBB and L is the set of least spenders in $(\mathbf{x}, \mathbf{p})$. The vector φ intuitively captures the potential at the allocation $(\mathbf{x}, \mathbf{p})$ during the run of our EF1+fPO Algorithm 1 starting with $\mathbf{x}^0$ as the initial allocation. That is, φ_i stores the utility of agent i the last time i was the LS in the run of Algorithm 1 starting from $(\mathbf{x}^0, \mathbf{p}^0)$, where $\mathbf{q}^0$ is the solution to the program (12) for the allocation $\mathbf{x}^0$.

We now describe the convex program (12), which for a given configuration $(\mathbf{x}, L, \varphi)$ finds a set of unique prices $\mathbf{p}$ s.t. $\sum_j p_j = n$, $(\mathbf{x}, \mathbf{p})$ is on MBB, and L is the set of least spenders in $(\mathbf{x}, \mathbf{p})$.

$$\begin{aligned} \text{maximize } & \prod_j p_j \\ \forall i \in N, \forall j \in M: \ & p_j \geq \lambda_i \cdot v_{ij} \\ \forall i \in N, \forall j \in M, \mathbf{x}_{ij} > 0: \ & p_j = \lambda_i \cdot v_{ij} \\ \forall i \in N: \ & p(\mathbf{x}_i) \geq \mu \\ \forall i \in L: \ & p(\mathbf{x}_i) = \mu \\ & \sum_j p_j = n \end{aligned} \tag{12}$$

The first two constraints capture the MBB condition; here, λ_i is a variable that represents the reciprocal of the MBB ratio of agent i. The variable μ captures the spending of the least spenders. Finally, the constraint $\sum_j p_j = n$ and the convexity of the objective $\prod_j p_j$ ensure that the set of prices respecting the MBB and LS constraints are unique and positive. The convex program can be solved in polynomial time since the number of constraints is polynomial in n and m.

Cost function. The cost of a configuration $(\mathbf{x}, L, \varphi)$ is a lexicographic cost function $\langle \delta(\mathbf{x}), \varphi \rangle$, where $\delta(\mathbf{x}) \in \{-1, 0, 1\}$. If $\mathbf{x}$ is EF1, then $\delta(\mathbf{x}) = 1$. If $\mathbf{x}$ is not EF1, then $\delta(\mathbf{x})$ is either -1 or 0, depending on whether the configuration is valid or not. Let $\mathbf{p}$ be the set of prices obtained by solving the program (12) for the allocation $\mathbf{x}$. If L does not equal the set of LS in $(\mathbf{x}, \mathbf{p})$, then $\delta(\mathbf{x}) = -1$. Otherwise if $v_i(\mathbf{x}_i) < \varphi_i$ for a LS i in $(\mathbf{x}, \mathbf{p})$, then also $\delta(\mathbf{x}) = -1$. These cases correspond to the configuration $(\mathbf{x}, L, \varphi)$ being invalid, i.e., L is not the correct set of LS, or φ_i cannot be the spending of i the last time i was an LS in a run of Algorithm 1 from $(\mathbf{x}^0, \mathbf{p}^0)$. In the other case, $(\mathbf{x}, L, \varphi)$ is valid and if $\mathbf{x}$ is not EF1, then $\delta(\mathbf{x}) = 0$. We note that the cost of configuration can be computed in polynomial time.

Neighborhood structure. A configuration $(\mathbf{x}, L, \varphi)$ with $\delta(\mathbf{x}) = 1$ has no neighbor. When $\delta(\mathbf{x}) = -1$, its neighbor is the allocation $(\mathbf{x}^0, L^0, 0^n)$, where $\mathbf{x}^0$ is the initial allocation and the set L^0 is the set of LS in $(\mathbf{x}^0, \mathbf{p}^0)$ for the prices $\mathbf{p}^0$ corresponding to $\mathbf{x}^0$. When $\delta(\mathbf{x}) = 0$, the configuration is valid and $\mathbf{x}$ is not EF1. We then compute $\mathbf{p}$ using (12) for the allocation $\mathbf{x}$, and run our Algorithm 1 with $(\mathbf{x}, \mathbf{p})$ as the initial configuration until the set of LS changes and we reach an allocation $(\mathbf{x}', \mathbf{p}')$, where the set of LS is L'. We then update the potential function to φ' whose i^{th} entry records the spending of agent i the last time i was an LS. The configuration $(\mathbf{x}', L', \varphi')$ is the neighbor of the $(\mathbf{x}, L, \varphi)$. From Lemma 4, we know the set of LS changes in $\mathsf{poly}(n, m)$ iterations, so the neighbor of a configuration can be computed in polynomial time.

Membership in PLS. Having defined the solution space, cost function, and the neighborhood structure of the local search problem Φ, we show that it is in PLS. PLS membership follows from demonstrating the following three polynomial time algorithms, which are implicit in the preceding paragraphs explaining the cost function and the neighborhood structure.

1. Algorithm A: Which outputs the initial allocation $(\mathbf{x}^0, \mathbf{p}^0)$.

2. Algorithm B: Which outputs the cost of a configuration.

3. Algorithm C: Which outputs a neighbor with strictly higher cost or is locally optimal.

Therefore, Φ is in PLS. Lastly, we argue that all the local optima of Φ correspond to EF1+fPO allocations. This is straightforward, since the local optima of Φ comprise of configurations $(\mathbf{x}, L, \varphi)$ where $\delta(\mathbf{x}) = 1$, which holds iff $\mathbf{x}$ is EF1. Since $\mathbf{x}$ is fPO from the definition of a configuration, $\mathbf{x}$ is EF1+fPO. We can, therefore, conclude:

Theorem 15. *The problem of computing an EF1+fPO allocation lies in* PLS*.*

The standard algorithm for PLS problems uses Algorithm A to find an initial solution and repeatedly uses Algorithms B and C to find neighbors with higher cost until a local optimum is reached. This standard algorithm when applied to Φ results in a sequence of allocations $(\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^T)$, where $\mathbf{x}^T$ is EF1+fPO, $T \leq \mathsf{poly}(n, m, U)$ and this sequence of allocations is a subsequence of allocations encountered in the run of Algorithm 1 starting from $(\mathbf{x}^0, \mathbf{p}^0)$.

Lastly, we remark that the above result does not show that the problem of computing an EF1+PO allocation is in PLS. In fact, since it is coNP-complete to check if a given allocation is PO, the problem of computing an EF1+PO allocation is not even in TFNP (unless $\mathsf{P} = \mathsf{NP}$).

8. Discussion

In this paper, we showed that an EF1+fPO allocation can be computed in pseudo-polynomial time, thus improving upon the result of Barman et al. (2018a). Our work establishes the polynomial time computability of EF1+PO allocations for two large non-trivial subclasses of instances: (i) k-ary valuations with constant k, and (ii) constant n (number of agents). These results are especially significant because polynomial-time computability was previously known only for the simple classes of binary or identical valuations. Moreover, computing the MNW allocation remains NP-hard for these classes, thus eliminating its use for efficient computation of an EF1+PO allocation. Further, these classes could be useful practically when the number of agents is small or the valuations are derived from asking the agents to rate items on a small scale. Our results also extend to the fairness notions of EQ1 and weighted EF1.

On the complexity front, we showed that computing an EF1+fPO lies in PLS. Settling the complexity of the EF1+fPO problem by designing a polynomial time algorithm or showing PLS-hardness remains a challenging open question. Finally, showing the existence of EF1+PO allocations for general additive instances of chores is another interesting research direction.

Acknowledgments

We would like to thank Ruta Mehta for discussion and insights regarding the proof of Theorem 15. Work on this paper is supported by NSF Grant CCF-1942321.

References

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

Amanatidis, G., Birmpas, G., Filos-Ratsikas, A., Hollender, A., & Voudouris, A. A. (2020). Maximum Nash welfare and other stories about EFX. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, (IJCAI)*, pp. 24–30.

Aziz, H., Lindsay, J., Ritossa, A., & Suzuki, M. (2022). Fair allocation of two types of chores..

Barman, S., Krishnamurthy, S. K., & Vaish, R. (2018a). Finding fair and efficient allocations. In *Proceedings of the 19th ACM Conference on Economics and Computation (EC)*, pp. 557–574.

Barman, S., Krishnamurthy, S. K., & Vaish, R. (2018b). Greedy algorithms for maximizing Nash social welfare. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, p. 7–13.

Bliem, B., Bredereck, R., & Niedermeier, R. (2016). Complexity of efficient and envy-free resource allocation: Few agents, resources, or utility levels. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, p. 102–108.

Brams, S., & Taylor, A. (1996). *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge University Press.

Brânzei, S., & Sandomirskiy, F. (2019). Algorithms for competitive division of chores. *CoRR*, *abs/1907.01766*.

Bredereck, R., Kaczmarczyk, A., Knop, D., & Niedermeier, R. (2019). High-multiplicity fair allocation: Lenstra empowered by n-fold integer programming. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, p. 505–523.

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

Caragiannis, I., Kurokawa, D., Moulin, H., Procaccia, A. D., Shah, N., & Wang, J. (2016). The unreasonable fairness of maximum Nash welfare. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*, p. 305–322.

Chakraborty, M., Igarashi, A., Suksompong, W., & Zick, Y. (2020). Weighted envy-freeness in indivisible item allocation. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, p. 231–239.

Cole, R., & Gkatzelis, V. (2015). Approximating the Nash social welfare with indivisible items. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing (STOC)*, p. 371–380.

Darmann, A., & Schauer, J. (2014). Maximizing Nash product social welfare in allocating indivisible goods. *SSRN Electronic Journal*, *247*.

de Keijzer, B., Bouveret, S., Klos, T., & Zhang, Y. (2009). On the complexity of efficiency and envy-freeness in fair division of indivisible goods with additive preferences. In Rossi, F., & Tsoukias, A. (Eds.), *Algorithmic Decision Theory*, pp. 98–110.

Duan, R., Garg, J., & Mehlhorn, K. (2016). An improved combinatorial polynomial algorithm for the linear arrow-debreu market. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 90–106.

Ebadian, S., Peters, D., & Shah, N. (2022). How to fairly allocate easy and difficult chores. In *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*.

Foley, D. (1967). Resource allocation and the public sector. *Yale Economic Essays*, *7*(1), 45–98.

Freeman, R., Sikdar, S., Vaish, R., & Xia, L. (2019). Equitable allocations of indivisible goods. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 280–286.

Freeman, R., Sikdar, S., Vaish, R., & Xia, L. (2020). Equitable allocations of indivisible chores. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, p. 384–392.

Garg, J., Hoefer, M., & Mehlhorn, K. (2017). Satiation in Fisher markets and approximation of Nash social welfare. *CoRR*, *abs/1707.04428*.

Garg, J., & Murhekar, A. (2021). On fair and efficient allocations of indivisible goods. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.

Garg, J., & Murhekar, A. (2023). Computing fair and efficient allocations with few utility values. *Theoretical Computer Science*, *962*, 113932.

Garg, J., Murhekar, A., & Qin, J. (2022). Fair and efficient allocations of chores under bivalued preferences. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*.

Garg, J., Murhekar, A., & Qin, J. (2023). New algorithms for the fair and efficient allocation of indivisible chores. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2710–2718.

Garg, J., Murhekar, A., & Qin, J. (2024). Weighted EF1 and PO allocations with few types of agents or chores. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*.

Halpern, D., Procaccia, A. D., Psomas, A., & Shah, N. (2020). Fair division with binary valuations: One rule to rule them all. In *Proceedings of Web and Internet Economics (WINE)*.

Johnson, D. S., Papadimitriou, C. H., & Yannakakis, M. (1988). How easy is local search?. *Journal of Computer and System Sciences*, *37*(1), 79 – 100.

Lee, E. (2015). APX-hardness of maximizing Nash social welfare with indivisible items. *Information Processing Letters*, *122*.

Lipton, R. J., Markakis, E., Mossel, E., & Saberi, A. (2004). On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce (EC)*, pp. 125–131.

Mas-Colell, A., Whinston, M., & Green, J. (1995). *Microeconomic Theory*. Oxford University Press.

Moulin, H. (2004). *Fair Division and Collective Welfare*. Mit Press. MIT Press.

Nguyen, T. T., & Rothe, J. (2014). Minimizing envy and maximizing average nash social welfare in the allocation of indivisible goods. *Discrete Applied Mathematics*, *179*, 54–68.

Othman, A., Sandholm, T., & Budish, E. (2010). Finding approximate competitive equilibria: Efficient and fair course allocation. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, p. 873–880.

Steinhaus, H. (1949). Sur la division pragmatique.. *Econometrica*, *17*(1), 315–319.

Varian, H. R. (1974). Equity, envy, and efficiency. *Journal of Economic Theory*, *9*(1), 63 – 91.

Walsh, T. (2020). Fair division: The computer scientist's perspective. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4966–4972.