Nearly Equitable Allocations Beyond Additivity and Monotonicity

SIDDHARTH BARMAN, Indian Institute of Science, India
UMANG BHASKAR, Tata Institute of Fundamental Research, India
YESHWANT PANDIT, Tata Institute of Fundamental Research, India
SOUMYAJIT PYNE, Tata Institute of Fundamental Research, India

Equitability (EQ) in fair division requires that items be allocated such that all agents value the bundle they receive equally. With indivisible items, equitability is not guaranteed when the allocation is complete. We hence consider a meaningful analog: equitability up to any item (EQX). EQX allocations exist for monotone, additive valuations. However, if (1) the agents' valuations are not additive, or (2) the set of indivisible items includes both goods and chores (positively and negatively valued items), then prior to this work, it was not known whether EQX allocations exist or not.

We study both the existence and efficient computation of EQX allocations. (1) For monotone valuations (not necessarily additive), we show that EQX allocations always exist, and for the large class of weakly well-layered valuations, these allocations can be found in polynomial time. Further, we prove that approximately EQX allocations can be computed efficiently under general monotone valuations. (2) For nonmonotone valuations, we show that an EQX allocation may not exist, even for two agents with additive valuations. Under some special cases, however, we establish the existence and efficient computability of EQX allocations. This includes the case of two agents with additive valuations where each item is either a good or a chore, and there are no mixed items. In general, we show that, under additive nonmonotone valuations, determining the existence of EQX allocations is weakly NP-hard for two agents and strongly NP-hard for more agents. We also give a pseudo-polynomial time algorithm for this problem for a constant number of agents.

JAIR Associate Editor: Piotr Skowron

JAIR Reference Format:
Siddharth Barman, Umang Bhaskar, Yeshwant Pandit, and Soumyajit Pyne. 2026. Nearly Equitable Allocations Beyond Additivity and Monotonicity. *Journal of Artificial Intelligence Research* 85, Article 2 (January 2026), 29 pages. DOI: 10.1613/jair.1.16658

1 Introduction

In the problem of fair division, a central planner (principal) is tasked with *fairly* partitioning a set of items among interested agents. If the items are indivisible, which is our focus, each item must be allocated integrally to an agent. Every agent i has a valuation function v_i that specifies agent i's value for each subset of items. Here, an item x could be a 'good', if every agent always values it positively, a 'chore', if every agent always values it negatively, or 'mixed', if across agents the value for item x can be both positive and negative.

What constitutes a fair allocation of items has no single answer. Over the years, various notions have been studied in depth (Moulin 2004). Possibly the most prominent among them are *envy-freeness* and *equitability*. An allocation is said to be envy-free if each agent prefers her own bundle of items to the bundle allocated to anyone else (Foley 1966). An allocation is said to be equitable if agents' values for their own bundles are the same and,

Authors' Contact Information: Siddharth Barman, ORCID: 0000-0001-9276-2181, barman@iisc.ac.in, Indian Institute of Science, Bengaluru, Karnataka, India; Umang Bhaskar, ORCID: 0009-0006-7939-7961, umang@tifr.res.in, Tata Institute of Fundamental Research, Mumbai, Maharashtra, India; Yeshwant Pandit, ORCID: 0009-0002-7800-2906, yeshwant.pandit@tifr.res.in, Tata Institute of Fundamental Research, Mumbai, Maharashtra, India; Soumyajit Pyne, ORCID: 0009-0009-5945-6268, soumyajit.pyne@tifr.res.in, Tata Institute of Fundamental Research, Mumbai, Maharashtra, India.

DOI: 10.1613/jair.1.16658

hence, the agents are equally content (Dubins and Spanier 1961). If the agents have identical valuations, then the two notions coincide.

Envy-freeness has received significant attention in classic fair division literature. It is known, for example, that for additive valuations over *divisible* goods, an allocation that maximizes the Nash social welfare (the product of the agents' values for their allocated bundles) is also envy-free (Varian 1974). The widely used platform Spliddit.org implements multiple methods to provide solutions for common fair division problems (Goldman and Procaccia 2015). The platform uses envy-freeness, in particular, as a fairness criterion for relevant applications.

Notably, equitability is a simpler construct to reason about, since it requires fewer comparisons. To test an allocation for equitability, we only need each agent's value for her own bundle, rather than every agent's value for every bundle.

Perhaps for this reason, equitable solutions are important in practical applications of fair division. Experimental studies have noted that, in specific fair-division settings, users tend to prefer equitable allocations over other notions of fairness (Herreiner and Puppe 2009).[1] Further, in bargaining experiments, equitability often plays a significant role in determining the outcome (Herreiner and Puppe 2010). Also in Spliddit.org, for fairly dividing rent among housemates, equitability was noted to be important as a refining criterion, following envy-freeness. The latest rent-division algorithm used in Spliddit.org computes solutions that satisfy this supporting objective (Gal et al. 2017).[2]

The real-world significance of equitability is further highlighted by considering divorce settlements. The two legal means of dividing property in the United States are *community property* and *equitable distribution* (Kagan 2021). In the community property rule, holdings are divided equally among the divorcing couple and, hence, the rule induces an envy-free division. Equitable distribution, on the other hand, takes into account various factors (such as the employability and financial needs of each party) for dividing the assets, and, in particular, makes inter-personal comparisons of value. Most states in the US follow equitable distribution, i.e., the courts divide assets and liabilities based on equitability. The definition of equitability in this situation is somewhat intuitive. However, the evidence clearly suggests that equitability is an important concept in practical situations.

Motivated by such considerations, the current work studies equitability (EQ), with the focus on allocating indivisible items. In the discrete fair division context, simple examples (with two agents and a single indivisible item) demonstrate that exactly equitable allocations may not exist. Hence, recent work has focused on relaxations. In fact, the same examples show that any multiplicative guarantees on the values of the agents, of the form $v_i(A_i) \geq \alpha \cdot v_j(A_j)$, for any finite α, are impossible. Hence, we consider additive guarantees. A compellingly strong relaxation is obtained by requiring that any existing inequality in agents' values is weakly reversed by the (hypothetical) removal of *any* good or chore in an appropriate manner. This relaxation is called equitability up to any item, EQX. Specifically, an allocation is said to be EQX if, whenever agent i has a lower value for her bundle, say A_i, than some other agent j for her bundle, say A_j (i.e., $v_i(A_i) < v_j(A_j)$), the removal of any positively-valued item (good) from A_j or the removal of any negatively-valued item (chore) from A_i ensures that the inequality is weakly reversed. Similarly, an allocation is EFX if, whenever agent i has a higher value for agent j's bundle than her own (i.e., $v_i(A_i) < v_i(A_j)$), this preference can be weakly reversed by removing a good from A_j or a chore from A_i. A weaker notion than EQX is EQ1, that requires that if $v_i(A_i) < v_j(A_j)$, then there exists either a positively-valued item (good) in $x \in A_j$ or a negatively-valued item (chore) $x \in A_i$ so that $v_i(A_i \setminus \{x\}) \geq v_j(A_j \setminus \{x\})$.

A basic question is then whether EQX allocations—or even EQ1 allocations—always exist. For the specific case of additively valued goods, EQX allocations are known to exist; this result is obtained via a greedy algorithm (Gourvès

[1] These prior papers refer to equitability as inequality inversion.

[2] Specifically, rent divisions that maximize the minimum value, called maximin solutions, are nearly equitable. The latest algorithm implemented in Spliddit.org finds rent divisions that satisfy envy-freeness and are also maximin.

et al. 2014).[3] In each iteration, the algorithm assigns an agent with the least bundle value her most valuable good from the remaining set of goods. Beyond this setting, however, this question has not been addressed in the literature. Our work addresses this notable gap. In particular, moving beyond monotone additive valuations, the current work establishes novel existential and algorithmic guarantees for EQX allocations. In this article, for succinctness, we use 'nonadditive' to denote the class of functions that includes both additive and nonadditive functions, and 'nonmonotone' to denote the class of functions that includes both monotone and nonmonotone functions. For monotone nonadditive valuations, for example, our work shows the universal existence of EQX allocations.

EQX requires allocations that are close to being equitable with a set-difference perspective, i.e., it requires that inequity can be resolved by removing any meaningful item. Hence, by design, this criterion is different from explicitly minimizing inequity in terms of valuations; the example below illustrates this distinction. In fact, such differences between standard fairness notions and optimization formulations arise in multiple other contexts, e.g., one can contrast MMS and the Santa Claus problem, or EFX and minimizing envy. Consider an instance with two agents and 103 items. Agent 1 values each item at 100, and agent 2 values each item at 1. Giving the first item to agent 1 and the remaining 102 items to agent 2 would put their values very close to each other (100 and 102), giving a nearly equitable allocation. This allocation, however, is not EQX (and not even EQ1). In fact, the only EQX allocation gives two items to agent 1 and the remaining 101 items to agent 2, giving values of 200 and 101 to the two agents — a much larger gap in the valuation.[4] Our work here focuses on EQX allocations.

1.1 Our Results and Techniques

Table 1. Our results for EQX allocations

	Monotone nonadditive valuations (§ 3)	Nonmonotone additive valuations (§ 4)
Existence of EQX allocations	EQX allocations always exist (Theorem 2)	• Exist for objective valuations with identical chores (Theorem 9) • Exist for objective valuations and a single chore (Theorem 10) • Do not exist for subjective valuations (Theorem 11)
Computational complexity	• Approximate EQX allocations are efficiently computable (Theorem 5) • Exact EQX allocations are efficiently computable in weakly well-layered valuations (Theorem 3)	• Efficiently computable for two agents with objective valuations (Theorem 6) • Weakly NP-hard for two agents with subjective valuations (Theorem 11) • Pseudopolynomial time algorithm for a fixed number of agents with subjective valuations (Theorem 13) • Strongly NP-hard if number of agents is not fixed (Theorem 14)

[3] EQX is termed as *near jealousy-freeness* in (Gourvès et al. 2014).
[4] Note that this example holds even for the objective of maximizing the minimum value — the objective is satisfied by the second allocation, but not the first allocation.

Under monotone, nondecreasing valuations, we establish (in Section 3):

- EQX allocations always exist, and can be computed in pseudo-polynomial time (Theorem 2). Our algorithm is based on a modification of an Add-and-Fix algorithm, proposed earlier for EFX under *identical* monotone nonincreasing valuations (Barman et al. 2023).
- Under *weakly well-layered* valuations, EQX allocations can be computed in polynomial time (Theorem 3). Weakly well-layered functions are an encompassing and natural class of valuations that include gross substitutes, weighted matroid-rank functions, budget-additive, well-layered, and cancelable valuations (Goldberg et al. 2023).
- For any $\varepsilon \in (0, 1)$, a $(1 - \varepsilon)$-approximately EQX allocation can be computed in time polynomial in $1/\varepsilon$ and the input size (Theorem 5).

Finding an EFX allocation is known to be hard (Plaut and Roughgarden 2020). Since the negative result holds even for two agents, with *identical* submodular valuations, the hardness also applies to EQX. This observation signifies that our polynomial-time algorithm for weakly well-layered valuations is the best possible, in the sense that such a positive result is unlikely for submodular valuations in general.

We then address nonmonotone additive valuations (Section 4). In comparison to monotone valuations, the results here are mixed and highlight a complicated landscape. Additive valuations are either objective — when an item either has nonnegative value for all agents, or nonpositive value for all agents — or subjective, when an item can have positive value for an agent and negative value for another.

- Existential guarantees:
  - For n agents with additive objective valuations where each chore c has the same value for the agents (i.e., $v_i(c) = v_j(c)$ for all agents i and j), we show—using the leximin++ ordering—that EQX allocations always exist (Theorem 9).
  - For n agents with additive objective valuations and a single chore, we show that EQX allocations exist (Theorem 10).
    The last result (Theorem 10) is technically challenging. It is obtained via a local search algorithm, that resolves EQX violations, whenever they arise, in a specific order. The analysis is subtle and needs to keep track of multiple progress measures, including the leximin++ value.
    Our work leaves open the question of whether EQX allocations exist for n agents with additive objective valuations. However for this case, we show that EQ1 allocations exist and can be computed efficiently (Proposition 7).
  - For agents with subjective valuations, EQX allocations may not exist (Theorem 11), validating our study of objective valuations.
- Computational results:
  - For two agents with additive objective valuations, we develop a polynomial-time algorithm for finding EQX allocations (Theorem 6). We note that for agents with *identical* additive valuations, Aziz and Rey (Aziz and Rey 2020) develop an efficient algorithm for finding EFX (and, hence, EQX) allocations. By the well-known cut-and-choose protocol, this gives an EFX algorithm for two nonidentical agents. However, cut-and-choose does not work for EQX and, hence, prior to our result there was no known algorithm for finding EQX allocation among two nonidentical agents.
    Many instances of fair division, in fact, consist of just two agents (such as in divorce settlement, or in the experiments of Gal et al. 2017), hence our result for two agents is of practical significance.
  - For a constant number of agents with nonmonotone, additive, subjective valuations, we provide a pseudo-polynomial time algorithm to determine whether an EQX allocation exists (Theorem 13).
  - For two agents with nonmonotone, additive, subjective valuations it is weakly NP-hard to determine whether an EQX allocation exists or not (Theorem 11).

– For arbitrary number of agents with nonmonotone, additive, subjective valuations determining whether there exists an EQX allocation is strongly NP-hard (Theorem 14).

Finally, in Appendix C, we show that all our results — with definitions modified appropriately — hold if we replace goods with chores, and vice versa. Thus, the results in Section 3 hold for monotone *nonincreasing* valuations (i.e., when all items are chores). Similarly, the results in Section 4 can be modified to show existence of EQX allocations for agents with objective valuation when each *good* has the same value for all agents, or if there is a single good.

Table 1 summarizes our results.

1.2 Additional Related Work

Equitability was first studied in the divisible setting of cake cutting. Here, equitable allocations for n agents exist (Cechlárová, Doboš, et al. 2013; Chèze 2017; Dubins and Spanier 1961) and, in particular, do not require additivity (Aumann and Dombb 2015; Bhaskar, Sricharan, et al. 2025). However, no finite algorithm can find an exactly equitable allocation (Ariel D Procaccia and Wang 2017), though approximately equitable allocations can be computed in near-linear time (Cechlárová and Pillárová 2012).

For indivisible items, EQX allocations were first shown to exist for monotone additive valuations (Freeman et al. 2020; Gourvès et al. 2014). This existential guarantee was obtained via an efficient greedy algorithm. For additive valuations that are strictly positive, allocations that are both EQX and Pareto optimal are known to exist (Freeman et al. 2019).

When the agents have identical valuations, EFX and EQX allocations coincide. Hence, under identical valuations, existential guarantees obtained for EFX allocations extend to EQX as well. In particular, for identical monotone (nondecreasing) valuations, EFX allocations were shown to exist using the leximin++ construct (Plaut and Roughgarden 2020). This work also showed that even for two agents with identical submodular valuations, finding an EFX allocation requires an exponential number of queries. This problem is also PLS-complete (Goldberg et al. 2023). For objective identical valuations, the existence of EFX allocations was shown through a modification of the leximin construct (Chen and Liu 2020). For additive identical valuations, EFX existence, as well as efficient computation, was obtained by Aziz and Rey (Aziz and Rey 2020).

Finally, a number of recent papers have also studied the loss of efficiency for equitable and near equitable allocations (Aumann and Dombb 2015; Bhaskar, Misra, et al. 2023; Caragiannis et al. 2012; Freeman et al. 2020, 2019; Sun et al. 2023).

2 Notation and Preliminaries

A fair division instance $(N, M, \mathcal{V})$ consists of a set $N = \{1, 2, \ldots, n\}$ of agents, a set M of indivisible items, with $m = |M|$, and a valuation function $v_i \in \mathcal{V}$ for each agent $i \in N$. The valuation $v_i : 2^M \to \mathbb{Z}$ specifies agent i's value for every subset of the items. We will assume, throughout, that $v_i(\emptyset) = 0$, and all the agents' values are integral.[5]

For notational convenience, for single items $x \in M$, we use $v_i(x)$ and $v_i(S \cup x)$ to denote $v_i(\{x\})$ and $v_i(S \cup \{x\})$, respectively. Let $V_{\max} := \max_{i \in [n], S \subseteq M} v_i(S)$. Given a valuation v and a subset $S \subseteq M$ of items, we say an item $x \in S$ is a *good* with respect to S if $v(S \cup x) > v(S)$, and a *chore* with respect to S if $v(S \cup x) < v(S)$ otherwise. In the case where $v(S \cup x) = v(S)$, whether x is a good or a chore will depend on the class of valuations studied, as described below.

[5]The integrality assumption holds without loss of generality for rational values, since we can multiply the rational numbers by the product of their denominators to get integral ones. Note that under such a scaling, the size of the input only increases polynomially.

- A valuation v is monotone nondecreasing if $v(S \cup x) \geq v(S)$ for all subsets $S \subseteq M$ and all items $x \in M$. In this case, all items are goods for this valuation. Similarly, valuation v is monotone nonincreasing if $v(S \cup x) \leq v(S)$ for all $S \subseteq M$ and $x \in M$ (and all items are chores in this case).
- Valuation v is additive if $v(S) = \sum_{x \in S} v(x)$ for all subsets $S \subseteq M$. In a fair division instance, additive valuations can be either *objective* or *subjective*. We say that a fair division instance has objective valuations if for each item $x \in M$ (i) either $v_i(x) \geq 0$ for all agents i, (ii) or $v_i(x) \leq 0$ for all agents i. Under (i), we say the item x is a good, and if case (ii) holds (and the inequality is strict for some agent i), then item x is a chore.
- Additive valuations are *subjective* if for some item x, there exist agents i and i' so that $v_i(x) > 0$ and $v_{i'}(x) < 0$. We then say that an item x is a good for agent i if $v_i(x) \geq 0$, and otherwise x is a chore for agent i. We will typically use g to denote a good and c to denote a chore.

An allocation $\mathcal{A} := (A_1, \ldots, A_n)$ is a partition of items M into n pairwise disjoint subsets. Here, subset of items $A_i \subseteq M$ is assigned to agent i (also called agent i's bundle). At times (such as when analyzing the interim allocations obtained by our algorithms), we may also consider partial allocations wherein not all items are assigned among the agents, i.e., for the pairwise disjoint bundles we have $\cup_{i=1}^{n} A_i \subsetneq M$. Given an allocation $\mathcal{A}$, we say an agent p is poorest if $v_p(A_p) = \min_{i \in N} v_i(A_i)$, and agent r is richest if $v_r(A_r) = \max_{i \in N} v_i(A_i)$.

Equitability and Envy-Freeness. An allocation $\mathcal{A}$ is equitable if $v_i(A_i) = v_j(A_j)$ for all agents $i, j \in N$. An allocation is said to be envy-free if $v_i(A_i) \geq v_i(A_j)$ for all agents $i, j \in N$. Hence, equity requires that all agents have equal value, while envy-freeness requires that each agent values her bundle at least that of any other agent. As mentioned, even in simple instances (with a single indivisible good and two agents), both equitable and envy-free allocations do not exist. Hence, we consider relaxations of these notions.

Specifically, an allocation $\mathcal{A} = (A_1, \ldots, A_n)$ is said to be equitable up to any item (EQX) if, for any pair of agents i, j, if $v_i(A_i) < v_j(A_j)$,
(1) for each good g for agent j, $v_j(A_j \setminus \{g\}) \leq v_i(A_i)$, and
(2) for each chore c for agent i, $v_i(A_i \setminus \{c\}) \geq v_j(A_j)$.

That is, in an EQX allocation, for each agent, the removal of any good assigned to her makes her a poorest agent, and the removal of any chore assigned to her must make her a richest agent.

Allocation $\mathcal{A} = (A_1, \ldots, A_n)$ is said to be envy-free up to any item (EFX) if, for all agents $i, j \in N$, we have $v_i(A_i) \geq v_i(A_j \setminus \{g\})$ for each good g for agent i and $v_i(A_i \setminus \{c\}) \geq v_i(A_j)$ for each chore c for agent i. Hence, from the perspective of agent i, the removal of any good from another agent's bundle should make it less valuable than i's own bundle, and the removal of any chore from agent i's bundle should make it more valuable than any other bundle.

Our results for monotone nondecreasing valuations (when all items are goods) are detailed in Section 3. Section 4 presents our results for nonmonotone additive valuations.

3 Monotone Valuations

In this section, all items have nonnegative marginal values. That is, $v_i(S \cup x) \geq v_i(S)$ for all items $x \in M$, agents $i \in N$, and subsets $S \subseteq M$. Hence, all items are goods in this section. We assume a standard value-oracle access to the valuations, that given any agent $i \in N$ and subset $S \subseteq M$, returns $v_i(S) \in \mathbb{Z}_{\geq 0}$.

For monotone valuations, we establish strong positive results towards the existence of EQX allocations. Our primary result shows that, for general monotone valuations, EQX allocations *always* exist, and for a broad class of valuations—termed *weakly well-layered* valuations—such fair allocations can be found in polynomial time.

Definition 1 ((Goldberg et al. 2023)). *A valuation function $v : 2^M \to \mathbb{Z}_{\geq 0}$ is said to be* weakly well-layered *if for any set $M' \subseteq M$ the sets $S_0, S_1, \ldots$ obtained by the greedy algorithm (that is, $S_0 = \emptyset$ and $S_i = S_{i-1} \cup \{x_i\}$, where $x_i \in \arg\max_{x \in M' \setminus S_{i-1}} v(S_{i-1} \cup x)$, for $i \leq |M'|$) are optimal, in the sense that $v(S_i) = \max_{S \subseteq M' : |S| = i} v(S)$ for all $i \leq |M'|$.*

Weakly well-layered valuations intuitively capture valuations where optimal sets can be obtained via a greedy algorithm. Several interesting and widely-studied classes of valuations are weakly well-layered, including gross substitutes (which include weighted matroid rank functions), budget-additive, well-layered, and cancelable valuations; Figure 1 in (Goldberg et al. 2023) is helpful in visualizing the relation between these classes. Notably, submodular functions are *not* weakly well-layered. This is an affirming observation, since it is known that, even for two agents with identical submodular valuations, obtaining an EFX (and, hence, EQX) allocation is PLS-complete *and* requires exponentially many value queries.

Our results are obtained through a modification and careful analysis of the Add-and-Fix algorithm, earlier used for obtaining EFX under *identical* cost functions (i.e., when all items are chores) (Barman et al. 2023).

The modified version of Add-and-Fix uses a greedy selection criterion; see Algorithm 1. The algorithm begins with an empty allocation. In each iteration, it identifies a poorest agent $p \in N$ in the current allocation. Then, agent p selects goods greedily from the unassigned ones. That is, from among the unassigned goods, p iteratively selects goods with *maximum marginal value*, until it is no longer the poorest agent. This is the Add phase in the algorithm (Lines 4 to 7). If after adding these goods, the allocation obtained is not EQX, this must be because of the goods assigned to agent p. In the Fix phase (Lines 8 to 10), violating goods are iteratively removed from agent p's bundle, until the allocation is EQX.

After each iteration, either the value of agent p increases, or the algorithm terminates (Claim 1). We differ from the original Add-and-Fix in two aspects: the original algorithm chose a richest agent in each iteration, since it dealt with chores, whereas we select a poorest agent. Secondly, and crucially for our results for efficient computation, the original algorithm assigned an arbitrary chore to the richest agent, while we select goods with maximum marginal contribution to the poorest agent. Also, note that the EFX guarantee in (Barman et al. 2023) is obtained for identical cost functions, whereas the EQX result here holds for monotone (not necessarily identical) valuations.

In our proofs, an iteration of the outer while-loop is called an outer iteration. Note that in every outer iteration, the allocation to every agent, other than p, remains unchanged. We will use the following claim.

Claim 1. *After each outer iteration, (i) either the value of the selected agent p strictly increases such that p is no longer the poorest agent (and the values of the other agents remain unchanged), or (ii) all the remaining unassigned goods are allocated to agent p and the algorithm terminates.*

Proof. At the beginning of the outer iteration, $v_p(A_p) \leq v_{p'}(A_{p'})$. The Add phase stops only when either all remaining goods are allocated to agent p or $v_p(A_p) > v_{p'}(A_{p'})$. Hence, at the end of the Add phase either $v_p(A_p) > v_{p'}(A_{p'})$, or all remaining goods are assigned to agent p and $v_p(A_p) \leq v_{p'}(A_{p'})$. In the latter case, the Fix phase is not executed, and the algorithm terminates as claimed. In the former case, the Fix phase is executed. Note, however, that a good $\widehat{g}$ is removed from agent p only if $v_p(A_p \setminus \{\widehat{g}\}) > v_{p'}(A_{p'})$. Hence, after termination of the Fix phase, we continue to have $v_p(A_p) > v_{p'}(A_{p'})$. Therefore, the value of agent p has strictly increased to above that of agent p', as claimed. □

Theorem 2. *Given any fair division instance with monotone valuations, Algorithm 1 computes an EQX allocation in pseudo-polynomial time.*

Proof. We first show that the algorithm terminates in pseudo-polynomial time. Claim 1 implies that the number of outer iterations is at most $\sum_{i \in N} v_i(M) \leq nV_{\max}$, where $V_{\max} := \max_i v_i(M)$. Each outer iteration

Algorithm 1 Greedy Add-and-Fix

Input: Fair division instance $(N, M, \mathcal{V})$ with value-oracle access to monotone, nondecreasing valuations.
Output: EQX allocation $\mathcal{A}$.

1: Initialize bundles $A_i = \emptyset$ for all agents i, and $U = M$ as the set of unassigned goods.
2: **while** $U \neq \emptyset$ **do**
3: Let $p \in \arg\min_{i \in N} v_i(A_i)$ and $p' \in \arg\min_{i \in N \setminus \{p\}} v_i(A_i)$.
{p and p' are a 'poorest' and 'second poorest' agent, respectively.}
{Add Phase: Lines 4 to 7}
4: **while** $v_p(A_p) \leq v_{p'}(A_{p'})$ **and** $U \neq \emptyset$ **do**
5: Let $g^* \in \arg\max_{g \in U}(v_p(A_p \cup g) - v_p(A_p))$.
6: Update $A_p \leftarrow A_p \cup \{g^*\}$ and $U \leftarrow U \setminus \{g^*\}$.
7: **end while**
{Fix Phase: Lines 8 to 10}
8: **while** there exists $\widehat{g} \in A_p$ such that $v_p(A_p \setminus \{\widehat{g}\}) > v_{p'}(A_{p'})$ **do**
9: Update $A_p \leftarrow A_p \setminus \{\widehat{g}\}$ and $U \leftarrow U \cup \{\widehat{g}\}$.
10: **end while**
11: **end while**
12: **return** Allocation $\mathcal{A} = (A_1, \ldots, A_n)$.

consists of an Add phase (in which at most m goods are included in agent p's bundle) and a Fix phase (in which at most m goods are removed from agent p's bundle). Each execution of these phases requires at most m calls to the value oracle to find the required goods g^* and $\widehat{g}$. Hence, it follows that the algorithm terminates in $O(m^2 n V_{\max})$ time.

Next, we show that the allocation computed by the algorithm is indeed EQX; our proof is via induction on the number of outer iterations. Initially, the allocation is empty, which is trivially EQX.

For the inductive step, fix any outer iteration and let p be the poorest agent selected in that iteration. Write $\mathcal{A} = (A_1, \ldots, A_n)$ for the allocation at the beginning of the outer iteration and $\mathcal{B} = (B_1, \ldots, B_n)$ for the allocation obtained after the outer iteration. Note that $B_i = A_i$ for all agents $i \neq p$. This observation and the induction hypothesis imply that any EQX violation must involve agent p. Further, Claim 1 gives us $v_p(B_p) \geq v_p(A_p)$.

To show that allocation $\mathcal{B}$ is EQX, we need to show that for any agent $i \in N$, the removal of any good $g \in B_i$ makes i a poorest agent. This condition holds—via the induction hypothesis—for all agents $i \neq p$; recall that $B_i = A_i$ for all $i \neq p$, and $v_p(B_p) \geq v_p(A_p)$.

For agent p, note that after the completion of the Fix phase, the removal of any good from p's bundle reduces her value to at most $v_{p'}(A_{p'}) = v_{p'}(B_{p'})$. Since p was the poorest and p' was the second poorest agent in allocation $\mathcal{A}$, this implies that the removal of any good from B_p would make agent p the poorest agent in $\mathcal{B}$ as well: $v_p(B_p \setminus \{g\}) \leq v_j(B_j)$ for all $j \in N$ and each good $g \in B_p$.

The theorem stands proved. □

3.1 Weakly Well-Layered Valuations

The following theorem asserts that for weakly well-layered valuations, Algorithm 1 computes an EQX allocation in polynomial time.

Theorem 3. *Given any fair division instance in which all the agents have monotone, weakly well-layered valuations, Algorithm 1 computes an EQX allocation in polynomial time.*

Proof. The monotonicity of agents' valuations ensures that the allocation returned by Algorithm 1 is EQX (Theorem 2). Hence, it remains to prove that, under weakly well-layered valuations, the algorithm terminates in polynomial time. Towards this, we will show that, in fact, the Fix phase never executes when all the valuations are weakly well-layered. Hence, in every outer iteration of Algorithm 1 the number of unassigned goods strictly decreases, and the algorithm terminates in polynomial time.

We will show that the Fix phase never executes via an inductive argument. In the base case—i.e., in the very first outer iteration—we have $A_i = \emptyset$ for all agents i. Now, for the first iteration, write S to denote the subset of goods assigned to the selected agent p in the Add phase. Further, let g^* denote the last good assigned in the Add phase. Then, by the loop-execution condition in Line 4, we have $v_p(S \setminus \{g^*\}) = 0$, since all other agents have value 0. Further, given that v_p is weakly well-layered and the set $S \setminus \{g^*\}$ is populated greedily, any set of goods of cardinality $|S| - 1$ has value 0. Hence, upon the removal of any good g from S, agent p has value 0 for the remaining subset, since it has size $|S| - 1$. Therefore, the Fix phase (see Line 8) will not execute in the first outer iteration.

For the inductive step, fix an outer iteration. Let $\mathcal{A} = (A_1, \ldots, A_n)$ be the allocation at the beginning of the iteration and $\overline{U} = M \setminus (\cup_i A_i)$ be the set of unassigned goods. For the poorest agent p selected in the iteration, consider the bundle A_p and write $\overline{S} \subseteq \overline{U}$ to denote the subset of goods assigned to p in the Add phase of the iteration. In addition, let $\overline{g} \in \overline{S}$ be the last good assigned in the Add phase. The following Claim asserts that all strict subsets $T \subsetneq \left(A_p \cup \overline{S}\right)$ have value $v_p(T) \leq v_p\left(\left(A_p \cup \overline{S}\right) \setminus \{\overline{g}\}\right)$.

Claim 4. *For weakly well-layered valuation v_p we have*

$$\left(A_p \cup \overline{S}\right) \setminus \{\overline{g}\} \in \arg\max_{X \subseteq A_p \cup \overline{S}} v_p(X).$$

We use Claim 4 to complete the inductive step. The execution criterion of the Add phase (Line 4) implies that before good $\overline{g}$ was included in the agent p's bundle, her value was at most $v_{p'}(A_{p'})$, i.e., $v_p\left(\left(A_p \cup \overline{S}\right) \setminus \{\overline{g}\}\right) \leq v_{p'}(A_{p'})$. Hence, via Claim 4, for every strict subset $T \subsetneq A_p \cup \overline{S}$ we have $v_p(T) \leq v_{p'}(A_{p'})$. The execution condition for the Fix phase will therefore not be satisfied. Since the Fix phase is never executed, the time taken by the algorithm to compute an EQX allocation is $O(m(n+m)) = O(m^2)$. This completes the proof of the theorem. □

Proof of Claim 4. The induction hypothesis implies that the Fix phase has not executed so far in the algorithm. Hence, during the previous iterations, the set of unassigned goods has decreased monotonically and the agents bundles, A_is, have increased monotonically. We can, hence, index the goods in A_p in the order in which they were included in the bundle, i.e., $A_p = \{g_1, g_2, \ldots, g_\ell\}$, where g_1 was the first good included in the bundle, and g_ℓ is the most recent. For each index $t \leq \ell$, write the prefix subset $G_t := \{g_1, \ldots, g_t\}$ and $G_0 := \emptyset$. Also, let U_t denote the set of unassigned goods from which good g_t was selected by agent p. As observed previously, we have $U_1 \supseteq U_2 \supseteq \ldots \supseteq U_\ell \supseteq \overline{U}$.

Therefore, for each index $t < \ell$, the following containment holds: $U_t \supseteq \overline{U} \cup \{g_t, g_{t+1}, \ldots, g_\ell\} = \overline{U} \cup (G_\ell \setminus G_{t-1})$. Note that good g_t was selected from U_t by agent p greedily to maximize the marginal contribution over G_{t-1}. That is, $g_t \in \arg\max_{g \in U_t} \left(v_p(G_{t-1} \cup g) - v_p(G_{t-1})\right)$ and, hence, the above-mentioned containment gives us

$$g_t \in \arg\max_{g \in \overline{U} \cup (G_\ell \setminus G_{t-1})} \left(v_p(G_{t-1} \cup g) - v_p(G_{t-1})\right). \tag{1}$$

By definition, we have $G_\ell = A_p$. Hence, Equation (1) implies that, in the set $\left(\overline{U} \cup A_p\right) \setminus G_{t-1}$, the good g_t has maximum marginal contribution over G_{t-1}:

$$g_t \in \underset{g \in \left(\overline{U} \cup A_p\right) \setminus G_{t-1}}{\arg\max} \left(v_p(G_{t-1} \cup g) - v_p(G_{t-1})\right). \tag{2}$$

Equation (2) holds for all indices $t \leq \ell$. Therefore, the greedy algorithm (as specified in Definition 1 and when executed for valuation v_p) would lead to subset $G_t = \{g_1, \ldots, g_t\}$, for each index $t \leq \ell$. Furthermore, with $G_\ell = A_p$, and given the selection criterion of the subset $\overline{S} \subseteq \overline{U}$ in the Add phase, the greedy algorithm when executed for $\left(|A_p \cup \overline{S}| - 1\right)$ steps over the goods in $A_p \cup \overline{U}$ would lead to the subset $(A_p \cup \overline{S}) \setminus \{\overline{g}\}$; recall that $\overline{g} \in \overline{S}$ was the last good included in p's bundle in the Add phase.

Hence, given that v_p is a weakly well-layered valuation (Definition 1), we obtain

$$\left(A_p \cup \overline{S}\right) \setminus \{\overline{g}\} \in \underset{X \subsetneq A_p \cup \overline{S}}{\arg\max}\ v_p(X).$$

The claim stands proved. □

3.2 Approximate EQX Allocations

As mentioned previously, for general monotone valuations, computing an EQX allocation is a PLS-hard problem. Complementing this hardness result, this section establishes that an approximately EQX allocation can be computed efficiently. In particular, for parameter $\varepsilon \in [0, 1]$, an allocation $\mathcal{A}$ is said to be an $(1-\varepsilon)$-EQX allocation if for every pair of agents $i, j \in N$ and for each good $g \in A_i$ we have $(1-\varepsilon)\, v_i(A_i \setminus \{g\}) \leq v_j(A_j)$. Hence, in an $(1-\varepsilon)$-EQX allocation, removing any good from any agent i's bundle brings down i's value to below $\frac{1}{1-\varepsilon}$ times the minimum. Also, note that $\varepsilon = 0$ corresponds to an exact EQX allocation.

We modify Algorithm 1 to obtain an approximately EQX allocation. We replace Lines 4 and 8 in Algorithm 1 with their approximate versions as follows:[6]

4: **while** $(1-\varepsilon)\, v_p(A_p) \leq v_{p'}(A_{p'})$ **and** $U \neq \emptyset$ **do**...

8: **while** there exists $\widehat{g} \in A_p$ such that $(1-\varepsilon) v_p(A_p \setminus \{\widehat{g}\}) > v_{p'}(A_{p'})$ **do**...

The following theorem provides our main approximation guarantee under monotone valuations. The proof of this result is similar to that of Theorem 2; see Appendix A.

Theorem 5. *Given parameter $\varepsilon \in (0, 1)$ and any fair division instance with monotone valuations, a $(1-\varepsilon)$-EQX allocation can be computed in $O\left(\frac{m^2 n}{\varepsilon} \log V_{\max}\right)$ time.*

Remark 1. *Note that the results obtained in this section for monotone nondecreasing valuations, also hold for monotone nonincreasing valuations (i.e., chores, rather than goods), with appropriate modifications to the algorithms. These modifications are introduced in Appendix C.*

4 Additive Objective Valuations

In this section and the next, we present our results for additive nonmonotone valuations. In this section, we present results for agents with additive objective valuations. We show that in multiple cases — for two agents; or when each chore has identical value for all agents; or there is a single chore — EQX allocations exist. Whether EQX allocations exist for the class of objective valuations is left as an open question.

In this section, we will use C to denote the set of chores and G to denote the set of goods. Since valuations are objective, $M = G \cup C$. Further, given an allocation $\mathcal{A}$, we say agent i has a goods violation if, for some good $g \in A_i$,

[6] Recall that Lines 4 and 8 in Algorithm 1 check the execution condition for the Add and Fix phases, respectively.

we have $v_i(A_i \setminus \{g\}) > \min_k v_k(A_k)$. Similarly, agent i has a chores violation if $v_i(A_i \setminus \{c\}) < \max_k v_k(A_k)$ for some chore $c \in A_i$.

4.1 Two Agents with Additive Objective Valuations

The section establishes that Algorithm 2 efficiently computes an EQX allocation between two (nonidentical) agents. A key property utilized in the analysis is that, if , in any iteration of Algorithm 2, item g^* is assigned to agent p (poor agent), then any chore c assigned previously (i.e., in an earlier iteration) to the other agent r (rich agent) must have greater absolute value. Similarly, if c^* is assigned to agent r in the considered iteration, any good g assigned previously to the other agent p must have greater absolute value. Using these bounds we show that the EQX criterion is maintained as an invariant as the algorithm iterates.

Algorithm 2 Two-Way Greedy Algorithm

Input: Instance $(N, M, \mathcal{V})$ with two agents and additive objective valuations.
Output: EQX allocation $\mathcal{A}$.

1: Initialize $\mathcal{A} = (\emptyset, \ldots, \emptyset)$ and item set $U := M$.
2: **while** $U \neq \emptyset$ **do**
3: Write $r \in \arg\max_{i \in N} v_i(A_i)$ and let p be the other agent ($p \neq r$).
4: Set item $g^* \in \arg\max_{g \in U \cap G} v_p(g)$ and item $c^* \in \arg\min_{c \in U \cap C} v_r(c)$. {$g^*$ is most valuable good for p and c^* is least valuable chore for r.}
5: **if** $|v_p(g^*)| > |v_r(c^*)|$ **then**
6: Update $A_p \leftarrow A_p \cup \{g^*\}$ and $U \leftarrow U \setminus \{g^*\}$.
7: **else**
8: Update $A_r \leftarrow A_r \cup \{c^*\}$ and $U \leftarrow U \setminus \{c^*\}$.
9: **end if**
10: **end while**
11: **return** Allocation $\mathcal{A} = (A_1, \ldots, A_n)$

Theorem 6. *Given any fair division instance $(N, M, \mathcal{V})$ with two agents that have additive objective valuations, Algorithm 2 computes an EQX allocation in polynomial time.*

Proof. We provide a proof of the theorem via induction on the number of allocated items. Initially, the algorithm starts with an empty allocation, which is trivially EQX. Now, fix any iteration of the algorithm and let $\mathcal{A}$ be the partial allocation maintained at the beginning of the iteration. Partial allocation $\mathcal{A}$ is EQX by the induction hypothesis. Also, for p and r as defined in the considered iteration, we have $v_p(A_p) \leq v_r(A_r)$.

Note that if, in the iteration, item g^* is assigned to agent p, then any chore c assigned previously (i.e., in an earlier iteration) to the other agent r must have greater absolute value. This follows from the algorithm's selection criterion (Line 5) and the fact that when chore c is assigned to agent r, the two agents p and r were similarly the poorer and the richer one, respectively. That is, $|v_p(g^*)| \leq |v_r(c)|$ for all chores $c \in A_r$. Similarly, if c^* is assigned to agent r in the considered iteration, any good g assigned previously to the other agent p must have greater absolute value, $|v_r(c^*)| \leq |v_p(g)|$ for all goods $g \in A_p$.[7]

In the considered iteration, either good g^* gets assigned to agent p or chore c^* is assigned to agent r. Next, we complete the inductive argument considering these cases.

[7]Recall that in the current context the two agents have objective valuations, i.e., the classification of items as goods or chores is consistent under the two valuations v_1 and v_2.

Case I*a : Good g^* is assigned to agent p and $v_p(A_p \cup g^*) \leq v_r(A_r)$.* By the induction hypothesis, for each chore $c \in A_p$, we have, $v_p(A_p \setminus \{c\}) \geq v_r(A_r)$. Hence, $v_p(A_p \cup g^* \setminus \{c\}) \geq v_r(A_r)$, i.e., there is no chores violation towards EQX. Also, by the induction hypothesis, for each good $g \in A_r$, it holds that $v_r(A_r \setminus \{g\}) \leq v_p(A_p) \leq v_p(A_p \cup g^*)$. Hence, in this case, EQX continues to hold even after the assignment of g^*.

Case I*b: Good g^* is assigned to agent p and $v_p(A_p \cup g^*) > v_r(A_r)$.* Since the agents select the goods greedily, each good in A_p has value at least $v_p(g^*)$. Hence, for each good $g \in A_p$, the following inequality holds: $v_p(A_p \cup g^* \setminus \{g\}) \leq v_p(A_p) \leq v_r(A_r)$. We obtain that there are no goods violation towards EQX. Next, recall that for each chore $c \in A_r$ the inequality $|v_r(c)| \geq v_p(g^*)$ holds. This inequality leads to the desired bound for each chore $c \in A_r$:

$$\begin{aligned} v_r(A_r \setminus \{c\}) &= v_r(A_r) + |v_r(c)| \geq v_p(A_p) + v_p(g^*) \\ &= v_p(A_p \cup g^*). \end{aligned}$$

Hence, there are no chores violations as well and, in the current case, the updated allocation is EQX.

Case II*a : Chore c^* is assigned to agent r and $v_p(A_p) \geq v_r(A_r \cup c^*)$.* Here, note that each chore c previously selected by agent r has absolute value at least $|v_r(c^*)|$. This bound gives us $v_p(A_p) \leq v_r(A_r) \leq v_r(A_r \cup c^* \setminus \{c\})$, for all chores $c \in A_r$. Hence, there are no chores violations in the updated allocation. Also, as mentioned previously, $|v_r(c^*)| \leq v_p(g)$ for all goods $g \in A_p$. Therefore, we obtain that the updated allocation upholds the EQX criterion with respect to goods' removal:

$$\begin{aligned} v_p(A_p \setminus \{g\}) &= v_p(A_p) - v_p(g) \leq v_r(A_r) + v_r(c^*) \\ &= v_r(A_r \cup c^*). \end{aligned}$$

Case II*b : Chore c^* is assigned to agent r and $v_p(A_p) < v_r(A_r \cup c^*)$.* The induction hypothesis gives us $v_r(A_r \setminus \{g\}) \leq v_p(A_p)$, for each good $g \in A_r$. Hence, there are no goods violations in the updated allocation: $v_r(A_r \cup c^* \setminus \{g\}) \leq v_r(A_r \setminus \{g\}) \leq v_p(A_p)$. The induction hypothesis also implies that $v_p(A_p \setminus \{c\}) \geq v_r(A_r)$ for each chore $c \in A_p$. Hence, there are no chores violations in the resulting allocation: $v_p(A_p \setminus \{c\}) \geq v_r(A_r) \geq v_r(A_r \cup c^*)$.

Overall, we obtain that EQX criterion is maintained as an invariant as the algorithm iterates. This completes the proof. □

We note that if we are satisfied with EQ1 allocations — a weaker relaxation of equitability, which requires that if $v_i(A_i) < v_j(A_j)$ then there exists *some* item $x \in A_i \cup A_j$ so that $v_i(A_i \setminus \{x\}) \geq v_j(A_j \setminus \{x\})$ — rather than EQX, then a simple greedy algorithm guarantees existence and efficient computation for any number of agents with additive objective valuations. The algorithm starts with an empty allocation, and orders the items arbitrarily. In each iteration, let $\mathcal{A}$ denote the partial allocation at the start, and p and r be the poorest and the richest agents in the partial allocation $\mathcal{A}$ respectively. Then, if the next unassigned item is a good g, it is assigned to agent p, otherwise (i.e., if the item is a chore c) it is assigned to agent r. We give the pseudocode as Algorithm 3. The following proposition asserts that the allocation returned by this algorithm is EQ1.

Proposition 7. *Given any fair division instance $(N, M, \mathcal{V})$ with additive objective valuations, Algorithm 3 computes an EQ1 allocation in polynomial time.*

Proof. The proof proceeds via induction on the number of items allocated. The algorithm begins with an empty allocation, which is trivially EQ1. This verifies the base case. For the inductive case, fix any iteration t of the algorithm and let $\mathcal{A}$ denote the partial allocation at the beginning of this iteration. By the induction hypothesis, $\mathcal{A}$ is EQ1. In the considered iteration, let p and r be a poorest and the richest agents respectively. We consider the following cases to prove the inductive case.

Algorithm 3 EQ1 Greedy Algorithm

Input: Instance $(N, M, \mathcal{V})$ with n agents and additive objective valuations.
Output: EQ1 allocation $\mathcal{A}$.

Initialize $\mathcal{A} = (\emptyset, \ldots, \emptyset)$. Let $M = \{j_1, j_2, \ldots, j_m\}$ be the set of items.
for $t \in [m]$ **do**
  Let $r \in \arg\max_{i \in N} v_i(A_i)$ and $p \in \arg\min_{i \in N} v_i(A_i)$.
  if $j_t \in G$ **then**
    Update $A_p \leftarrow A_p \cup \{j_t\}$. {Item j_t is a good}
  else
    Update $A_r \leftarrow A_r \cup \{j_t\}$. {Item j_t is a chore}
  end if
end for
return Allocation $\mathcal{A} = (A_1, \ldots, A_n)$

Case I*: Item j_t is assigned to agent p.*
In this case, item j_t must be a good. Consider the set $N \setminus \{p\}$ of agents. Since the bundles of agents in this set remain unchanged, by the induction hypothesis, the allocation restricted to this set of agents is EQ1. Now, consider any agent $i \in N \setminus \{p\}$. If $v_i(A_i) < v_p(A_p \cup j_t)$, then $v_i(A_i) \geq v_p(A_p)$. Otherwise, by the induction hypothesis, there exists some item $x \in A_i \cup A_p$ so that $v_p(A_p \setminus \{x\}) \geq v_i(A_i \setminus \{x\})$ which also implies $v_p(A_p \cup \{j_t\} \setminus \{x\}) \geq v_i(A_i \setminus \{x\})$. Thus, the allocation obtained after the good j_t is assigned to agent p is EQ1.
Case II *: Item j_t is assigned to agent r.*
In this case, item j_t must be a chore. As in the earlier case, according to the induction hypothesis, the allocation restricted to the set $N \setminus \{r\}$ of agents is EQ1. Consider any agent $i \in N \setminus \{r\}$. If $v_i(A_i) > v_r(A_r \cup j_t)$, then $v_i(A_i) \leq v_r(A_r)$. Otherwise, by the induction hypothesis, there exists some item $x \in A_i \cup A_r$ so that $v_i(A_i \setminus \{x\}) \geq v_r(A_r \setminus \{x\})$ which further implies $v_i(A_i \setminus \{x\}) \geq v_r(A_r \cup \{j_t\} \setminus \{x\})$. Thus, in this case also the allocation obtained is EQ1. □

4.2 Identically-Valued Chores

This section shows that EQX allocations are guaranteed to exist when the agents have additive, objective valuations and they value the chores c identically, i.e., $v_i(c) = v_j(c)$ for all agents $i, j \in N$.

As noted, C denotes the set of chores, and G is the set of goods. For each chore $c \in C$, we will write $v_c < 0$ to denote the common value of the chore among the agents.

For intuition for the proof, consider an allocation $\mathcal{A} = (A_1, \ldots, A_n)$ that is not EQX. There are two possibilities. There could be a chores violation — there exist agents i and j and a chore $c \in A_i$ such that $v_i(A_i \setminus \{c\}) < v_j(A_j)$. Here c is called the violating chore. Or, there could be a goods violation — there exist agents i and j and a good $g \in A_j$ such that $v_i(A_i) < v_j(A_j \setminus \{g\})$. Here g is a violating good.

Our proof is based on the observation that transferring c to A_j in case of a chores violation and g to A_i under a goods violation leads to a lexicographic improvement in the tuple of values. The lexicographic order here additionally incorporates the sizes of the assigned bundles and the agents' indices for tie-breaking.

Specifically, for any allocation $\mathcal{X} = (X_1, \ldots, X_n)$, we define permutation (ordering) $\sigma^{\mathcal{X}} \in \mathbb{S}_n$ [8] over the n agents such that

(i) Agents i with lower values, $v_i(X_i)$, receive lower indices in $\sigma^{\mathcal{X}}$.
(ii) Among agents with equal values, agents i with lower bundle sizes, $|A_i|$, receive lower indices in $\sigma^{\mathcal{X}}$.

[8] $\mathbb{S}_n$ is the set of all permutations of set of agents N

(iii) Then, among agents with equal values and equal number of items, we order by index i.

Plaut and Roughgarden (Plaut and Roughgarden 2020) used a similar idea in their proof of the existence of EFX allocations for *identical* valuations. They noted, via an example, that the lexicographic order over just the tuple of values does not suffice and a strengthening that incorporates bundle sizes is required. In particular, they introduced the $\preceq_{++}$ comparison operator, which we detail in Algorithm 4. Unlike their work, we use the $\preceq_{++}$ operator to obtain existential guarantees for EQX allocations even when the valuations are nonidentical over the goods (but identical for chores).

Note that if two agents have the same value and the same number of items, then Plaut and Roughgarden's results hold for any arbitrary but consistent tie-breaking between agents. In the definition of $\preceq_{++}$ (see Algorithm 4), we use the index of each agent $i \in [n]$ as a tie-breaker to compare the allocations to two agents that are otherwise identical (i.e., both agents have the same value for their own bundles and the same number of items).

Algorithm 4 $\preceq_{++}$ comparison operator

Input: Allocations $\mathcal{A}$ and $\mathcal{B}$.
Output: TRUE if $\mathcal{A} \preceq_{++} \mathcal{B}$, else return FALSE.

Let $\sigma^{\mathcal{A}}$ and $\sigma^{\mathcal{B}}$ be the defined permutations of the agents associated with allocations $\mathcal{A}$ and $\mathcal{B}$, respectively.
for all $\ell \in [n]$ **do**
  Set $i = \sigma^{\mathcal{A}}(\ell)$ and set $j = \sigma^{\mathcal{B}}(\ell)$. {$i$ and j are the ℓth agents in the permutations.}
  if $v_i(A_i) \neq v_j(B_j)$ **then**
    return $v_i(A_i) < v_j(B_j)$.
  else if $|A_i| \neq |B_j|$ **then**
    return $|A_i| < |B_j|$.
  else if $i \neq j$ **then**
    return $i < j$.
  end if
end for
return TRUE. {Each agent has the same value and the same number of items in each allocation.}

Theorem 8 (Theorem 4.1, (Plaut and Roughgarden 2020)). *The comparison operator $\preceq_{++}$ induces a total order on the set of all allocations.*

We say that an allocation $\mathcal{A}$ is a leximin++ allocation if $\mathcal{B} \preceq_{++} \mathcal{A}$ for all allocations $\mathcal{B}$. Theorem 8 ensures that a leximin++ allocation is guaranteed to exist. However, computing a leximin++ allocation is intractable (Plaut and Roughgarden 2020).

The following theorem then establishes the existential guarantee for EQX allocations in the case of identically-valued chores.

Theorem 9. *In a fair division instance if the agents have additive, objective valuations and they value the chores identically, then the leximin++ allocation is EQX.*

Proof. Write $\mathcal{A}$ to denote a leximin++ allocation in the given instance; such an allocation necessarily exists (Theorem 8). Assume, towards a contradiction, that allocation $\mathcal{A}$ is not EQX. Then, under $\mathcal{A}$, we either have a goods violation or a chores violation.

We consider these two cases separately and show that, in both the cases, allocation $\mathcal{A}$ is not leximin++. In particular, the allocation $\mathcal{B}$ obtained by transferring the violating good or the violating chore satisfies $\mathcal{A} \preceq_{++} \mathcal{B}$.

Goods Violation: Here, let i and j be the agents with the property that $v_i(A_i) < v_j(A_j \setminus \{g\})$ for a violating good $g \in A_j$. Now, consider the allocation $\mathcal{B} = (B_1, \ldots, B_n)$ obtained by transferring g to agent i. That is, let $B_i = A_i \cup \{g\}$ and $B_j = A_j \setminus \{g\}$; the bundles of all the other agents remain unchanged. Note that $v_i(B_i) \geq v_i(A_i)$ and $|B_i| > |A_i|$, and since g was a violating good, $v_j(B_j) > v_i(A_i)$.

Let ℓ be the index of agent i in the permutation $\sigma^{\mathcal{A}}$, i.e., $\sigma^{\mathcal{A}}(\ell) = i$. Note first that the first $(\ell - 1)$ agents remain unchanged in $\sigma^{\mathcal{A}}$ and $\sigma^{\mathcal{B}}$ – these are agents with values, bundle sizes, and indices—in that order—lower than that of i. Now, if the ℓth agent in permutation $\sigma^{\mathcal{B}}$ is agent i itself, then $\mathcal{A} \preceq_{++} \mathcal{B}$. This follows from the fact that $v_i(B_i) \geq v_i(A_i)$ and $|B_i| > |A_i|$. In case the ℓth agent in $\sigma^{\mathcal{B}}$ is j, then again $\mathcal{A} \preceq_{++} \mathcal{B}$; recall that $v_j(B_j) > v_i(A_i)$. Finally, if the ℓth agent, $\sigma^{\mathcal{B}}(\ell)$, is neither i nor j, then this agent was placed after agent i in $\sigma^{\mathcal{A}}$. Since the bundle assigned to the agent $\sigma^{\mathcal{B}}(\ell)$ is unchanged, it has either a higher value, more items, or a larger index than i. In this case as well, $\mathcal{A} \preceq_{++} \mathcal{B}$.

Chores Violation: Here, let i and j be the agents with the property that $v_i(A_i \setminus \{c\}) < v_j(A_j)$ for some chore $c \in A_i$. Now, consider the allocation $\mathcal{B} = (B_1, \ldots, B_n)$ obtained by transferring chore c to agent j. That is, $B_i = A_i \setminus \{c\}$ and $B_j = A_j \cup \{c\}$. Recall that the common value of the chore v_c is negative. Hence, the following strict inequality holds $v_i(B_i) > v_i(A_i)$. Also, since c is a violating chore, we have $v_j(A_j) > v_i(A_i) - v_c$. The last inequality reduces to $v_j(B_j) = v_j(A_j) + v_c > v_i(A_i)$.[9]

Again, let ℓ be the index of agent i in the permutation $\sigma^{\mathcal{A}}$, i.e., $\sigma^{\mathcal{A}}(\ell) = i$. As before, the first $(\ell - 1)$ agents remain unchanged in $\sigma^{\mathcal{A}}$ and $\sigma^{\mathcal{B}}$. If the ℓth agent in $\sigma^{\mathcal{B}}$ is agent i itself, then, using the above-mentioned strict inequality, $v_i(B_i) > v_i(A_i)$, we obtain $\mathcal{A} \preceq_{++} \mathcal{B}$. In case the ℓth agent in $\sigma^{\mathcal{B}}$ is j, then again $\mathcal{A} \preceq_{++} \mathcal{B}$, since $v_j(B_j) > v_i(A_i)$. Finally, if the ℓth agent is neither i nor j, then this agent was placed after agent i in $\sigma^{\mathcal{A}}$. Since the bundle assigned to this agent is unchanged, it has either a higher value, more items, or a larger index than i. Overall, we obtain the desired lexicographic dominance, $\mathcal{A} \preceq_{++} \mathcal{B}$. □

4.3 Additive Goods and a Single Chore

We now show that in instances with additive objective valuations and a single chore c, EQX allocations exist. Note first that in the presence of even a single non-identical chore, the leximin++ allocation may no longer be EQX (Appendix B). Instead, our proof of existence is based on a careful analysis of a version of local search, and keeping track of the movement of the single chore.

The local search algorithm (Algorithm 5) proceeds as follows. We start off with an arbitrary allocation of items to agents (the algorithm assigns everything to agent 1, but any initial allocation also works). Write $\mathcal{A}$ to denote the current allocation. Further, let p be the agent that appears first in the permutation $\sigma^{\mathcal{A}}$ (as defined in the previous subsection) and r be the agent that appears last. Note that p and r are a poorest and a richest agent, respectively. If $\mathcal{A}$ is not EQX, then there must be either a goods violation or a chores violation.

If there is a goods violation in $\mathcal{A}$, then there exists an agent i and good $g \in A_i$ with the property that $v_i(A_i \setminus \{g\}) > v_p(A_p)$. In our algorithm, we resolve the goods violation by transferring good g to agent p. The algorithm resolves all goods violations, before addressing the chores violation. Now, if there is a chores violation, then for some agent i and the chore $c \in A_i$ it holds that $v_i(A_i \setminus \{c\}) < v_r(A_r)$. The algorithm resolves the chores violation by transferring chore c to the richest agent r. The algorithm repeats these steps—resolving all goods violations and then the chores violation—until there are no more violations.

By design, the algorithm terminates only when the maintained allocation is EQX. To prove that the algorithm indeed terminates, we need the following notation. For an allocation $\mathcal{A}$, we denote by $\kappa(\mathcal{A})$ the agent holding the unique chore c, i.e., $c \in A_{\kappa(\mathcal{A})}$. We call the value $v^+(\mathcal{A}) := v_{\kappa(\mathcal{A})}(A_{\kappa(\mathcal{A})} \setminus \{c\})$ as the *cutoff value* of the allocation $\mathcal{A}$ and of the agent $\kappa(\mathcal{A})$.

[9]This is where we require that the chores be identical; the value of c does not change when it is transferred from agent i to agent j.

Algorithm 5 Algorithm for single-chore setting

Input: Fair division instance $(N, M, \mathcal{V})$ with additive, objective valuations and a single chore c
Output: EQX allocation $\mathcal{A}$

Initialize $A_1 = M$ and $A_i = \emptyset$ for all agents $i \neq 1$.
while $\mathcal{A} = (A_1, \ldots, A_n)$ is **not** EQX **do**
  $p = \sigma^{\mathcal{A}}(1)$. {$p$ is the first agent according to $\sigma^{\mathcal{A}}$.}
  while there exists agent $i \in N$ and good $g \in A_i$ such that $v_i(A_i \setminus \{g\}) > v_p(A_p)$ **do**
    Update $A_p \leftarrow A_p \cup \{g\}$ and $A_i \leftarrow A_i \setminus \{g\}$.
    Update $p = \sigma^{\mathcal{A}}(1)$.
  end while {After resolving all goods violations, check for chores violation.}
  $r \leftarrow \sigma^{\mathcal{A}}(n)$. {$r$ is the last agent according to $\sigma^{\mathcal{A}}$.}
  if $v^+(\mathcal{A}) < v_r(A_r)$ **then**
    Update $A_{\kappa(\mathcal{A})} \leftarrow A_{\kappa(\mathcal{A})} \setminus \{c\}$ and $A_r \leftarrow A_r \cup \{c\}$. {$\kappa(\mathcal{A})$ is the agent holding c.}
  end if
end while
return $\mathcal{A}$

In case of a chores violation we have $v_{\kappa(\mathcal{A})}(A_{\kappa(\mathcal{A})} \setminus \{c\}) < v_r(A_r)$. Write $\mathcal{B}$ to denote the allocation after transferring chore c to agent r. Note that such a chore transfer increases the cutoff value: under allocation $\mathcal{A}$, the cutoff value is $v^+(\mathcal{A}) = v_{\kappa(\mathcal{A})}(A_{\kappa(\mathcal{A})} \setminus \{c\})$ and, under the updated allocation $\mathcal{B}$, it is $v^+(\mathcal{B}) = v_r(A_r)$. In particular, $v^+(\mathcal{A}) < v^+(\mathcal{B})$. In our analysis, we will keep track of two progress measures separately: the lexicographic value of the allocation (according to the order $\preceq_{++}$ defined in Section 4.2) and the cutoff value of the allocation. We will show, in particular, that the cutoff value of the allocation is nondecreasing between successive chores violations, and strictly increases whenever a chores violation is resolved. Whenever a goods violation is resolved, we obtain a lexicographic improvement in the allocation; a chores violation may however result in a lexicographic decrease. Since both the cutoff value and the lexicographic value can only increase a finite number of times, the local search algorithm must terminate in finite time. Formally,

Theorem 10. *Given any fair division instance with additive objective valuations and a single chore, Algorithm 5 terminates in finite time and returns an EQX allocation.*

Proof. The outer while-loop terminates only when the current allocation is EQX. Hence, if the algorithm terminates, it must return an EQX allocation. It remains to show that the algorithm terminates.

Firstly, consider the inner while-loop, which resolves goods violations. It follows from the proof of Theorem 9 that resolving a goods violation leads to a lexicographic improvement. That is, if $\mathcal{A}$ is the allocation before the inner while-loop executes and $\mathcal{B}$ is the allocation after the inner while-loop terminates, then $\mathcal{A} \preceq_{++} \mathcal{B}$. In particular, $\min_i v_i(A_i) \leq \min_i v_i(B_i)$.

Now consider two successive executions of Line 9. Let $\mathcal{A}'$ be the allocation just before the first execution of Line 9, $\mathcal{A}$ be the allocation after the chores violation is resolved, and $\mathcal{B}$ be the allocation just before the second execution of Line 9 (i.e., after the inner while loop terminates). We claim that $v^+(\mathcal{A}') < v^+(\mathcal{B})$.

To prove the claim, we first show that $v^+(\mathcal{A}') < v^+(\mathcal{A})$. Towards this, note that if there is a chores violation in $\mathcal{A}'$, then the chore is moved from agent $\kappa(\mathcal{A}')$ to agent $\kappa(\mathcal{A})$. Further, $v_{\kappa(\mathcal{A}')}(A'_{\kappa(\mathcal{A}')} \setminus \{c\}) < v_{\kappa(\mathcal{A})}(A'_{\kappa(\mathcal{A})})$, since $\kappa(\mathcal{A})$ is agent r in $\mathcal{A}'$. Then, $v^+(\mathcal{A}') < v^+(\mathcal{A})$ by definition of the cutoff value.

We now show that $v^+(\mathcal{A}) \leq v^+(\mathcal{B})$. Write $k := \kappa(\mathcal{A})$, and note that agent k holds the chore in $\mathcal{A}$, $\mathcal{B}$, and all the interim allocations during the execution of the inner while loop. We consider two cases.

In the first case, agent k is a poorest agent in $\mathcal{A}$, after receiving the chore. Then, as mentioned earlier, $\mathcal{A} \preceq_{++} \mathcal{B}$ and, in particular, the minimum value is nondecreasing in the inner while loop. Hence, $v_k(A_k) \leq v_k(B_k)$. Since the cutoff value is obtained by removing the chore from A_k and B_k, it follows that $v^+(\mathcal{A}) \leq v^+(\mathcal{B})$.

In the second case, agent k is not a poorest agent in $\mathcal{A}$. In this case, we will show that either k does not lose any goods in the inner while loop, in which case clearly $v_k(A_k) \leq v_k(B_k)$, and hence $v^+(\mathcal{A}) \leq v^+(\mathcal{B})$. If k does lose a good is some iteration, we show that it must have received a good in an earlier iteration. But to receive a good, it must have been the poorest agent in this iteration, which allows us to use the earlier proof for when agent k is a poorest agent.

Formally, since the previous allocation $\mathcal{A}'$ had no goods violations, removal of any good $g \in A'_k$ made k the poorest agent in $\mathcal{A}'$. This property continues to hold after the chore is transferred to agent k. That is, $v_k(A_k \setminus \{g\}) \leq v_i(A_i)$ for each good $g \in A_k$ and all agents $i \neq k$.

Let T denote the number of iterations of the inner while-loop when it updates allocation $\mathcal{A}$ to allocation $\mathcal{B}$. Write $\mathcal{A}^i$ to denote the allocation after the ith execution of the inner while loop. Then $\mathcal{A} = \mathcal{A}^0$, and $\mathcal{B} = \mathcal{A}^T$.

Clearly, if the value of agent k does not decrease in the T iterations of the inner while loop, then $v^+(\mathcal{A}) \leq v^+(\mathcal{B})$ and we are done. So suppose that agent k's value decreases in the inner while loop. Let t be the first iteration of the inner while loop where a good is removed from agent k's bundle (this is a necessary condition for agent k's value to decrease). However as we have noted, in allocation $\mathcal{A}$, removing any good from agent k's bundle made it the poorest agent. If agent k's bundle is unchanged prior to this execution, i.e., $A_k^{t-1} = A_k$, then since the minimum value is nondecreasing through the inner while loop, removing any good from agent k's bundle would still make it the poorest agent, and hence the condition in the inner while loop would not be met for agent k. Hence, since a good g is removed from agent k in the tth while loop, we get that $A_k^{t-1} \supsetneq A_k$, and there must exist some $t' < t$ such that a good was added to agent k's bundle in execution t'.

Now let $t' < t$ be the first execution of the inner while loop where a good is added to agent k's bundle. For a good to be added to agent k's bundle, agent k must be the poorest agent prior to the addition, i.e., $v_k(A_k^{t'-1}) = \min_i v_i(A_i^{t'-1})$. Note that this is the first modification of agent k's bundle by the inner while loop, hence $A_k^{t'-1} = A_k$. But we know that the minimum value is nondecreasing in the inner while loop. Hence, we obtain that

$$v_k(B_k) \geq \min_i v_i(B_i) \geq \min_i v_i(A_i^{t'-1}) = v_k(A_k^{t'-1}) = v_k(A_k).$$

This completes the proof of the claim $v^+(\mathcal{A}) \leq v^+(\mathcal{B})$ since agent k holds the chore in allocations $\mathcal{A}$ and $\mathcal{B}$.

Therefore, we obtain that between two successive executions of Line 9, the cutoff value of the allocation must strictly increase. This can clearly happen a finite number of times. After the last execution of Line 9, after resolving goods violations in the inner while-loop (which must terminate, since lexicographic improvements can only occur a finite number of times), there cannot be any chores violations, and the outer while-loop must terminate and return an EQX allocation. □

5 Additive Subjective Valuations

We now focus on additive subjective valuations. Here, our results are mostly negative. We show that even for two agents with subjective valuations, EQX allocations may not exist. Furthermore, EQX allocations can be hard to compute even if they do exist. Some of our results extend beyond additive valuations, however, in this section, we will conform to additive valuations for ease of exposition (and use subjective to mean additive subjective valuations).

It is known that an EQX allocation can be computed in polynomial time if all the agents have identical additive valuations (Aziz and Rey 2020). Complementing this positive result, we next show that an EQX allocation may not exist among two agents that have nonidentical subjective valuations. The instance demonstrating this

nonexistence of EQX allocations is given in Table 2. The computational hardness of determining existence is established via a reduction from the Partition problem in the following theorem. We note that the nonexistence and computational hardness hold even for *normalized* valuations — when $v_i(M)$ is equal for all agents i, where M is the set of all items.

Theorem 11. *An EQX allocation may not exist for two agents with subjective normalized valuations. Further, in such instances, it is weakly* NP*-hard to determine whether an EQX allocation exists or not.*

Proof. We show nonexistence of EQX allocations in the instance given in Table 2. Assume, towards a contradiction, that there is an EQX allocation $\mathcal{A} = (A_1, A_2)$ in the specified instance. Note that, here, under EQX, for $i = 1$ and $i = 2$, agent i cannot receive two items that are both goods for i. Otherwise, removing the good, besides x_3, from i's bundle would not drive $v_i(A_i)$ below the other agent's value. That is, for $\mathcal{A}$ to be EQX, it must hold that $|A_1 \cap \{x_1, x_3\}| \leq 1$ and $|A_2 \cap \{x_2, x_3\}| \leq 1$.

Table 2. Example showing nonexistence of EQX allocations in Theorem 11.

	x_1	x_2	x_3
Agent 1	+1	−1	+100
Agent 2	−1	+1	+100

Therefore, if $x_3 \in A_1$, then $x_1 \in A_2$. Now, note that x_1 is a chore for agent 2 and even after removing it from A_2, we have $v_2(A_2 \setminus \{x_1\}) \leq 1$. In addition, the containment $x_3 \in A_1$ ensures $v_1(A_1) \geq 99$. These inequalities imply that even after the removal of item x_1 (which is a chore for agent 2) from A_2, agent 2's value does not exceed $v_1(A_1)$. That is, the EQX condition cannot hold if $x_3 \in A_1$.

On the other hand, if $x_3 \in A_2$, then we have $x_2 \in A_1$. Item x_2 is a chore for agent 1, and in the current case the following inequalities hold: $v_1(A_1 \setminus \{x_2\}) \leq 1$ and $v_2(A_2) \geq 99$. Therefore, again, allocation $\mathcal{A}$ does not uphold the EQX criterion. Overall, we obtain that the instance does not admit an EQX allocation.

To establish NP-hardness, we provide a reduction from Partition, known to be NP-complete. An instance of Partition consists of a set of t positive integers $a_1, \ldots, a_t$. The problem is to determine if there exists a subset $S \subseteq [t]$ such that $\sum_{i \in S} a_i = \sum_{i \notin S} a_i$.

Given an instance of Partition, we construct a fair division instance as follows. There are two agents 1 and 2, and $t + 2$ items. The first two items, x_1 and x_2, are similar to the first two items in Table 2. That is, item x_1 has value +1 for agent 1 and −1 for agent 2. Item x_2 has value −1 for agent 1 and +1 for agent 2. The remaining items, $g_1, \ldots, g_t$, are all identical goods, with item g_i possessing value $2a_i$ for both agents (i.e., twice the value of corresponding integer in the Partition instance).

We show that the Partition instance has a solution iff the constructed fair division instance admits an EQX allocation. Table 3 depicts the reduction.

Table 3. The reduction from Partition for Theorem 11.

	x_1	x_2	g_1	...	g_t
Agent 1	+1	−1	$2a_1$	...	$2a_t$
Agent 2	−1	+1	$2a_1$	...	$2a_t$

Forward Direction: Suppose there exists $S \subseteq [t]$ such that $\sum_{i \in S} a_i = \sum_{i \notin S} a_i$. Then, consider the allocation that assigns agent 1 items x_1, x_2, and all goods with indices in S. This results in an equitable (hence, an EQX) allocation

in which both the agents' values are equal to $\sum_{i \in M} a_i$.

Reverse Direction: In any allocation $\mathcal{A} = (A_1, A_2)$ of the fair division instance, and no matter how x_1 and x_2 are allocated, the absolute difference in value $|v_1(A_1) - v_2(A_2)|$ must be even. For the reverse direction of the reduction, assume that the constructed fair division instance admits an EQX allocation $\mathcal{A} = (A_1, A_2)$. We first claim that $|v_1(A_1) - v_2(A_2)| \leq 1$. To see this, assume without loss of generality that $x_1 \in A_1$. Then since $\mathcal{A}$ is EQX, $v_1(A_1) \leq v_2(A_2) + 1$. If $x_2 \in A_2$, then $v_2(A_2) \leq v_1(A_1) + 1$ as well, and hence the claim holds. If $x_2 \in A_1$, then $v_1(A_1) \geq v_2(A_2) - 1$, and again the claim holds. Since $|v_1(A_1) - v_2(A_2)|$ must be even, we get that

$$v_1(A_1) = v_2(A_2)\,. \tag{3}$$

Further, note that there are three possible cases when considering the assignments of the items x_1 and x_2 between the two bundles A_1 and A_2:

(i) Both the items are assigned to the same agent. Since $v_1(\{x_1, x_2\}) = v_2(\{x_1, x_2\}) = 0$, in this case we have $v_1(A_1) = v_1(A_1 \setminus \{x_1, x_2\})$ and $v_2(A_2) = v_2(A_2 \setminus \{x_1, x_2\})$. Using equation (3), we further obtain

$$v_1(A_1 \setminus \{x_1, x_2\}) = v_2(A_2 \setminus \{x_1, x_2\}) \tag{4}$$

(ii) The items x_1, x_2 are split between the bundles with $x_1 \in A_1$ and $x_2 \in A_2$. In this case, $v_1(A_1) = v_1(A_1 \setminus \{x_1\}) + 1$ and $v_2(A_2) = v_2(A_2 \setminus \{x_2\}) + 1$. Again, the equality $v_1(A_1 \setminus \{x_1, x_2\}) = v_2(A_2 \setminus \{x_1, x_2\})$ holds.

(iii) The items x_1, x_2 are split between the bundles with $x_1 \in A_2$ and $x_2 \in A_1$. Here, $v_1(A_1) = v_1(A_1 \setminus \{x_2\}) - 1$ and $v_2(A_2) = v_2(A_2 \setminus \{x_1\}) - 1$. This leads to the same equality obtained in the previous cases: $v_1(A_1 \setminus \{x_1, x_2\}) = v_2(A_2 \setminus \{x_1, x_2\})$.

That is, equality (4) holds in all three cases. Hence, the goods in the set $A_1 \setminus \{x_1, x_2\}$ correspond to elements $S \subseteq [t]$ with the property that

$$\begin{aligned} \sum_{i \in S} a_i &= \frac{1}{2} v_1(A_1 \setminus \{x_1, x_2\}) \\ &= \frac{1}{2} v_2(A_2 \setminus \{x_1, x_2\}) && \text{(via (4))} \\ &= \sum_{j \notin S} a_j. \end{aligned}$$

Therefore, the set S is a solution for Partition instance. This establishes the reverse direction of the reduction and completes the proof of the theorem. □

5.1 Pseudo-polynomial Time Algorithm for a Fixed Number of Agents with Additive Valuations

We now give a dynamic programming algorithm that operates in pseudo-polynomial time and determines the existence of an EQX allocation in a given fair division instance with a constant number of agents with additive subjective valuations. For ease of exposition, we will limit our discussion to the case of two agents. The algorithm and analysis can be easily extended to a constant number of agents.

Let us assume that the items in M are labelled $x_1, \ldots, x_m$. Let $H \coloneqq \max_{i \in N} \max_{x_j \in M} |v_i(x_j)|$. The dynamic programming algorithm has two components: table-filling and table-checking. The table-filling component of algorithm creates a binary-valued table $A[j, w_1, w_2, l_1, h_1, l_2, h_2]$. An entry $A[j, w_1, w_2, l_1, h_1, l_2, h_2] = 1$ if there exists an allocation of the first j items such that agent 1's value for her bundle is w_1, agent 2's value for her bundle is w_2, the value of a minimum-valued good in agent 1's bundle is l_1 while that for agent 2 is l_2, and the

value of a maximum-valued chore in agent 1's bundle is h_1 while that for agent 2 is h_2. For an agent i, we thus have $w_i \in [-mH, mH],\ l_i \in [0, H],\ h_i \in [-H, 0]$ and the parameter H is as defined before. Note that for values of the parameters outside the range, we have $A[j, w_1, w_2, l_1, h_1, l_2, h_2] = 0$.

Once we have values for all the entries of the table, checking the existence of an EQX allocation is a simple task — it is carried out by the table-checking component of algorithm. This component examines $A[m, w_1, w_2, l_1, h_1, l_2, h_2]$ for all values of w_i, l_i and h_i for $i \in \{1, 2\}$. For each table entry with $A[m, w_1, w_2, l_1, h_1, l_2, h_2] = 1$, the component verifies the EQX conditions: If $w_2 > w_1$ (agent 2 values her bundle higher than agent 1 values her own), then we check whether $w_2 - l_2 \leq w_1$ (removal of a minimum-valued good from agent 2's bundle causes inequity to vanish) and $w_1 - h_1 \geq w_2$ (removal of a maximum-valued chore from agent 1's bundle causes inequity to vanish) are satisfied. Similarly, if $w_1 \geq w_2$, we check whether the inequalities $w_1 - l_1 \leq w_2$ and $w_2 - h_2 \geq w_1$ are satisfied. If the algorithm finds a positive entry in the table for which the EQX conditions hold, then it declares that an EQX allocations exists. Otherwise, if no such entry is found, the algorithm declares that the given instance does not admit an EQX allocation.

Algorithm 6 Table-filling

Input: Instance $(N, M, \mathcal{V})$ with two agents and additive subjective valuations.
Output: Binary-valued table A

1: Set all entries of A to 0 initially
2: **if** $(v_1(x_1) \geq 0)$ then $A(1, v_1(x_1), 0, v_1(x_1), 0, 0, 0) = 1$ {x_1 is assigned to agent 1}
else $A(1, v_1(x_1), 0, 0, v_1(x_1), 0, 0) = 1$
3: **if** $(v_2(x_1) \geq 0)$ then $A(1, 0, v_2(x_1), 0, 0, v_2(x_1), 0) = 1$ {x_1 is assigned to agent 2}
else $A(1, 0, v_2(x_1), 0, 0, 0, v_2(x_1)) = 1$
4: **for** $j \in [2, m]$
for $i \in \{1, 2\},\ w_i \in [-mH, mH],\ l_i \in [0, H],\ h_i \in [-H, 0]$

$$f_1 = \begin{cases} \vee_{l_1' \in [l_1, H]} A[j-1, w_1 - v_1(x_j), w_2, l_1', h_1, l_2, h_2] & \text{if } v_1(x_j) \geq 0,\ v_1(x_j) = l_1 \\ A[j-1, w_1 - v_1(x_j), w_2, l_1, h_1, l_2, h_2] & \text{if } v_1(x_j) \geq 0,\ v_1(x_j) > l_1 \\ 0 & \text{if } v_1(x_j) \geq 0,\ v_1(x_j) < l_1 \\ \vee_{h_1' \in [-H, h_1]} A[j-1, w_1 - v_1(x_j), w_2, l_1, h_1', l_2, h_2] & \text{if } v_1(x_j) < 0,\ v_1(x_j) = h_1 \\ A[j-1, w_1 - v_1(x_j), w_2, l_1, h_1, l_2, h_2] & \text{if } v_1(x_j) < 0,\ v_1(x_j) < h_1 \\ 0 & \text{if } v_1(x_j) < 0,\ v_1(x_j) > h_1 \end{cases}$$

$$f_2 = \begin{cases} \vee_{l_2' \in [l_2, H]} A[j-1, w_1, w_2 - v_2(x_j), l_1, h_1, l_2', h_2] & \text{if } v_2(x_j) \geq 0,\ v_2(x_j) = l_2 \\ A[j-1, w_1, w_2 - v_2(x_j), l_1, h_1, l_2, h_2] & \text{if } v_2(x_j) \geq 0,\ v_2(x_j) > l_2 \\ 0 & \text{if } v_2(x_j) \geq 0,\ v_2(x_j) < l_2 \\ \vee_{h_2' \in [-H, h_2]} A[j-1, w_1, w_2 - v_2(x_j), l_1, h_1, l_2, h_2'] & \text{if } v_2(x_j) < 0,\ v_2(x_j) = h_2 \\ A[j-1, w_1, w_2 - v_2(x_j), l_1, h_1, l_2, h_2] & \text{if } v_2(x_j) < 0,\ v_2(x_j) < h_2 \\ 0 & \text{if } v_2(x_j) < 0,\ v_2(x_j) > h_2 \end{cases}$$

$$A[j, w_1, w_2, l_1, h_1, l_2, h_2] = f_1 \vee f_2$$

5: **return** A.

Algorithm 7 Table-checking

Input: Instance $(N, M, \mathcal{V})$ with two agents and additive subjective valuations.
Output: TRUE if an EQX allocation exists, FALSE otherwise.

```
A =Table-filling((N, M, V))
for all entries with A[m, w_1, w_2, l_1, h_1, l_2, h_2] = 1 do
  if w_1 < w_2 then
    if w_2 − l_2 ≤ w_1 and w_1 − h_1 ≥ w_2 then
      return TRUE
    end if
  else
    if w_1 − l_1 ≤ w_2 and w_2 − h_2 ≥ w_1 then
      return TRUE
    end if
  end if
end for
return FALSE
```

Lemma 12. *Given any fair division instance involving two agents with additive subjective valuations, Algorithm 6 correctly computes the table $A[j, w_1, w_2, l_1, h_1, l_2, h_2]$ for all values of the parameters j, w_i, l_i and h_i for $i \in \{1, 2\}$.*

Proof. We use induction on the number of items involved in an allocation.

Let us consider the base case. Here, we consider allocations involving only one item x_1. An allocation can have either item x_1 assigned to the agent 1 or item x_1 assigned to the agent 2. Consider the case in which item x_1 is assigned to the agent 1. If item x_1 is a good for agent 1 then we have $A[1, v_1(x_1), 0, v_1(x_1), 0, 0, 0] = 1$. Otherwise, if x_1 is a chore for agent 1, we have $A[1, v_1(x_1), 0, 0, v_1(x_1), 0, 0] = 1$. Similarly if item x_1 is assigned to the agent 2, we get either $A[1, 0, v_2(x_1), 0, 0, v_2(x_1), 0] = 1$ or $A[1, 0, v_2(x_1), 0, 0, 0, v_2(x_1)] = 1$ depending on whether x_1 is a good or a chore for agent 2. Since these are the only two feasible allocations of x_1, for other values of the parameters, the algorithm initializes $A[1, w_1, w_2, l_1, h_1, l_2, h_2] = 0$. Thus, the algorithm computes $A[1, w_1, w_2, l_1, h_1, l_2, h_2]$ correctly for all values of the parameters and the base case is verified.

For the inductive case, let us assume that the algorithm correctly computes $A[j-1, w_1, w_2, l_1, h_1, l_2, h_2]$ for all values of the parameters. To compute $A[j, w_1, w_2, l_1, h_1, l_2, h_2]$, as in the base case, the algorithm considers four cases: (i) the item x_j is assigned to agent 1, and is a good for agent 1(ii) the item x_j is assigned to agent 1, and is a chore for agent 1 (iii) the item x_j is assigned to agent 2, and is a good for agent 2 and (iv) the item x_j is assigned to agent 2, and is a chore for agent 2. Consider case (i), the other cases are symmetric. In case (i), we have three possibilities: (1) $v_1(x_j) = l_1$, i.e., the value of x_j for agent 1 is equal to the desired value of minimum valued good in agent 1's bundle (2) $v_1(x_j) > l_1$ and (3) $v_1(x_j) < l_1$.

(1) If $v_1(x_j) = l_1$ then the algorithm checks if there exists an allocation of the first $j-1$ items, in which $w_1 - v_1(x_j)$ is agent 1's value for her bundle and the desired value of minimum valued good in agent 1's bundle is greater than or equal to l_1 with values for all other parameters being same. By induction hypothesis, we know that the value of $A[j-1, w_1, w_2, l_1, h_1, l_2, h_2]$, for all values of the parameters are computed correctly. Thus, the algorithm correctly updates the value of f_1 in this case.

(2) Suppose $v_1(x_j) > l_1$, then the algorithm checks if there exists an allocation of first $j-1$ items and $w_1 - v_1(x_j)$ as agent 1's value for her bundle with values for other parameters being same. Again, by induction hypothesis we can conclude that the algorithm updates f_1 correctly.

(3) In case $v_1(x_j) < l_1$, it's not possible to get an allocation satisfying the parameter l_1. Hence, the algorithm assigns 0 to f_1.

Thus, the algorithm updates f_1 correctly. Since the other cases are symmetric, we also conclude that the algorithm updates f_2 correctly. We then take the OR ($\vee$) of the variables f_1 and f_2 to obtain the value of $A[j, w_1, w_2, l_1, h_1, l_2, h_2]$. Thus, the value of $A[j, w_1, w_2, l_1, h_1, l_2, h_2]$ is computed correctly by the algorithm. □

Theorem 13. *Given any fair division instance involving two agents with additive subjective valuations, Algorithm 7 correctly determines the existence of an EQX allocation in pseudo-polynomial time.*

Proof. In the first step, Algorithm 7 invokes the table-filling algorithm which returns a binary-valued table A. We saw in Lemma 12 that this table is computed correctly by the table-filling algorithm.

Algorithm 7 then verifies EQX conditions for all entries $A[m, w_1, w_2, l_1, h_1, l_2, h_2] = 1$, i.e., if $w_2 > w_1$ (agent 2 values her bundle higher than agent 1 values her own) then it checks whether $w_2 - l_2 \leq w_1$ (removal of a minimum-valued good from agent 2's bundle causes inequity to vanish) and $w_1 - h_1 \geq w_2$ (removal of a maximum-valued chore from agent 1's bundle causes inequity to vanish) are satisfied. Similarly if $w_1 > w_2$, it checks whether $w_1 - l_1 \leq w_2$ and $w_2 - h_2 \geq w_1$ are satisfied. The algorithm returns TRUE if EQX conditions are satisified for some w_i, l_i and h_i for $i \in \{1, 2\}$ and returns FALSE otherwise. This completes the proof of correctness.

We now show that the Algorithm 7 terminates in pseudo-polynomial time. The algorithm first invokes table-filling algorithm which takes $O(m^3H^7)$ time to complete its execution. It then verifies EQX conditions and decides the existence of EQX allocation in the given instance. This part of the execution takes $O(m^2H^6)$ time. Thus, the overall execution time of the algorithm is $O(m^3H^7 + m^2H^6)$. □

Note that if there are $c > 2$ agents, the table looks like $A[j, w_1, w_2, \ldots, w_c, l_1, h_1, l_2, h_2, \ldots, l_c, h_c]$ and the runtime will be $O(m^{c+1}H^{3c+1} + m^cH^{3c})$, where $O(m^{c+1}H^{3c+1})$ is the time taken by the table-filling algorithm, and $O(m^cH^{3c})$ is the time required to verify the EQX conditions.

5.2 Strong NP-hardness for Arbitrary Number of Agents with Additive Valuations

We now show that if the number of agents is part of the input, determining whether there exists an EQX allocation is strongly NP-hard. The result is established using a reduction from 3-Partition which is known to be strongly NP-hard.

Theorem 14. *Given any fair division instance with additive subjective valuations, determining whether there exists an EQX allocation is strongly* NP*-hard.*

Proof. The proof proceeds via a reduction from 3-Partition, which is known to be strongly NP-hard. In 3-Partition, we are given a multiset of $3n$ positive integers $S = \{a_1, a_2, \ldots, a_{3n}\}$ where $n \in \mathbb{N}$, the sum of all integers in S is nT, i.e., $\sum_{i=1}^{3n} a_i = nT$ and each integer a_i satisfies $a_i \in \left(\frac{T}{4}, \frac{T}{2}\right)$. The objective is to find a partition of S into n subsets $S_1, S_2, \ldots, S_n$ such that the sum of integers in each subset is T. Note that the condition $a_i \in \left(\frac{T}{4}, \frac{T}{2}\right)$ ensures that each subset S_i is of cardinality 3.

Given an instance of 3-Partition, we construct a fair division instance as follows. There are $n + 1$ agents $1, 2, \ldots, n + 1$ and $3n + 2$ items $x_1, x_2, \ldots, x_{3n+2}$. Agents $1, \ldots, n$ are called "original" agents, and items $x_1, \ldots, x_{3n}$ are called "original" items. Also, let $\Delta \gg 10nT$ be a sufficiently large positive integer.

Now, we define the valuations for the agents. We have $v_i(x_j) = a_j$ for each $i \in [n]$ and $j \in [3n]$. Hence, the items $x_1, \ldots, x_{3n}$ are identically valued by the agents $1, \ldots, n$ and their values correspond to the integers in S. Items x_{3n+1} and x_{3n+2} are also identically valued by the agents in $[n]$, with values $\frac{\Delta}{10}$ and $-\frac{\Delta}{10}$, respectively: $v_i(x_{3n+1}) = \frac{\Delta}{10}$ and $v_i(x_{3n+2}) = -\frac{\Delta}{10}$, for each $i \in [n]$.

The agent $n+1$ values the items $x_1, \ldots, x_{3n}$ at $-\Delta$, i.e., $v_{n+1}(x_j) = -\Delta$ for each $j \in [3n]$. Further, agent $n+1$ values the remaining items x_{3n+1} and x_{3n+2} at T and 1, respectively. The valuations for the constructed fair division instance are depicted in Table 4.

Table 4. Fair division instance for establishing strong NP-hardness

	x_1	x_2	$\cdots$	x_{3n}	x_{3n+1}	x_{3n+2}
Agent 1	a_1	a_2	$\cdots$	a_{3n}	$\frac{\Delta}{10}$	$-\frac{\Delta}{10}$
Agent 2	a_1	a_2	$\cdots$	a_{3n}	$\frac{\Delta}{10}$	$-\frac{\Delta}{10}$
.	.	.	$\cdots$	.	.	.
.	.	.	$\cdots$	.	.	.
Agent n	a_1	a_2	$\cdots$	a_{3n}	$\frac{\Delta}{10}$	$-\frac{\Delta}{10}$
Agent $n+1$	$-\Delta$	$-\Delta$	$\cdots$	$-\Delta$	T	1

We show that the 3-Partition instance has a solution if and only if the constructed fair division instance admits an EQX allocation.

Forward Direction: Suppose $S_1, S_2, \ldots, S_n$ is a solution of the 3-Partition instance. Then an EQX allocation $A = (A_1, \ldots, A_{n+1})$ can be constructed as follows. If integer a_j belongs to the subset S_i then add item x_j to the agent i's bundle A_i. Assign items x_{3n+1} and x_{3n+2} to the agent $n+1$. In the resulting allocation, agents in $[n]$ have equal value T for their bundles while agent n+1 values her bundle at T+1. This is clearly an EQX allocation.

Reverse Direction: Let $\mathcal{A} = (A_1, \ldots, A_n)$ be an EQX allocation. We first claim that no original item is assigned to agent $n+1$. Assume, towards a contradiction, that some original item x_j is assigned to the last agent $n+1$. Since $v_{n+1}(x_j) = -\Delta$, we have $v_{n+1}(A_{n+1}) < -\frac{\Delta}{2}$. In fact, if along with x_j, another original item, say x_k, is assigned to agent $n+1$, then even after the removal of x_k we would have $v_{n+1}(A_{n+1} \setminus \{x_k\}) < -\frac{\Delta}{2}$. On the other hand, for each original agent $i \in [n]$, it must hold that $v_i(A_i) \geq -\frac{\Delta}{10}$. Hence, if A_{n+1} contains two (or more) original items, then the EQX condition cannot hold for agent $n+1$. This observation implies that, in allocation $\mathcal{A}$, all the original items, besides x_j, must have been assigned among the original agents. Therefore, there exists an agent $i \in [n]$ whose bundle A_i is of size at least two. Further, considering the original agents' valuations, we obtain that A_i must contain a good x and $v_i(A_i \setminus \{x\}) \geq -\frac{\Delta}{10}$. However, this inequality and the bound $v_{n+1}(A_{n+1}) < -\frac{\Delta}{2}$ contradict the fact that $\mathcal{A}$ is an EQX allocation. Therefore, by way of contradiction, we have that no original item is assigned to agent $n+1$.

We next show that both items x_{3n+1} and x_{3n+2} must be assigned to agent $n+1$.

(i) Assume, towards a contradiction, that $x_{3n+2} \notin A_{n+1}$. Since the original agents are symmetric, we can, without loss of generality, further assume that $x_{3n+2} \in A_1$. We consider two cases, depending on if $x_{3n+1} \in A_1$ or not. In the first case (i.e., if $x_{3n+1} \in A_1$), we have $A_{n+1} = \emptyset$ and, hence, $v_{n+1}(A_{n+1}) = 0$. Here, for EQX to hold, it must hold that removing a good from any other agent $\{1, \ldots, n\}$ should reduce her value to be at

most zero. In particular, each of the original agents can get at most one original item. However, such a containment requirement cannot hold, since there are $3n$ original items (and agent $n+1$ cannot get any original items). Therefore, we obtain a contradiction, when both $x_{3n+2} \in A_1$ and $x_{3n+1} \in A_1$.
It remains to address the case wherein $x_{3n+2} \in A_1$ and $x_{3n+1} \notin A_1$. Here, $v_1(A_1) < 0$, and, for EQX to hold, it must hold that removing a good from any remaining agent $i \in \{2, \ldots, n+1\}$ should leave agent i with a negatively valued bundle. However, this requirement is not met, since some agent $i \in \{2, \ldots, n+1\}$ will possess good x_{3n+1}. If $x_{3n+1} \in A_{n+1}$, then this is the only item held by $n+1$, and removing it gives agent $n+1$ zero value. For the other agents, every item except x_{3n+2} is a good and thus removing x_{3n+1} similarly does not leave the agent with a negatively valued bundle.Thus, we get a contradiction, and hence $x_{3n+2} \in A_{n+1}$.

(ii) Next, assume, towards a contradiction, that $x_{3n+1} \notin A_{n+1}$. Then, $A_{n+1} = \{x_{3n+2}\}$ and we have $v_{n+1}(A_{n+1}) = 1$. Further, items $x_1, \ldots, x_{3n+1}$ must be allocated to the n original agents, who have value at least 1 for each of the items. Hence, in such a case, some original agent must receive at least 4 of these items. Removing an item still leaves her with a bundle of value at least 3 and, hence, $\mathcal{A}$ cannot be EQX. Therefore, we get that $x_{3n+1} \in A_{n+1}$.

The analysis above implies that, in EQX allocation $\mathcal{A}$, items $x_1, \ldots, x_{3n}$ must be assigned to the agents in $[n]$, while items x_{3n+1} and x_{3n+2} must be assigned to the agent $n+1$.

Furthermore, each agent in $[n]$ must possess a bundle of value at least T, since $v_{n+1}(A_{n+1} \setminus \{x_{3n+2}\}) = v_{n+1}(x_{3n+1}) = T$. Since $\sum_{i=1}^{3n} a_i = nT$, each agent must get value exactly T from her bundle. Hence, an EQX allocation induces a solution of the 3-Partition instance. □

6 Conclusion and Future Work

Our work resolves fundamental questions regarding the existence of EQX allocations. We present sweeping positive results when all the items are goods *or* all items are chores; this includes universal existence of EQX allocations under general, monotone valuations and an accompanying pseudo-polynomial time algorithm. For monotone valuations, we also provide a fully polynomial-time approximation scheme (FPTAS) for finding approximately EQX allocations. In addition, we show that under weakly well-layered valuations EQX allocations can be computed efficiently.

For mixed items (goods and chores), we show that EQX allocations may not exist, and show both hardness results and polynomial time algorithms for determining the existence of an EQX allocation. For additively-valued goods and chores, our results present a mixed picture: existence and efficient computation for two agents, and existence either when each chore or each good has the same value among the agents, or if there is a single chore or a single good.

A number of significant open questions remain. First, the existence of EQX allocations under objective, additive valuations remains unresolved. Second, efficient algorithms for computing EQX allocations in this case (or even for goods and a single chore, or identical chores) appear challenging. The practical significance of equitability also motivates the study of truthful mechanisms for obtaining EQX allocations. Finally, Proposition 7 shows that EQ1 allocations exist for agents with additive objective valuations (and can be obtained in polynomial time). The existence and efficient computation of EQ1 allocations in more general valuation classes remains an open problem, and may prove useful in resolving the open questions for EQX allocations as well.

Acknowledgments

We thank Rohit Vaish for useful discussions and suggestions regarding this paper. UB, YP, and SP acknowledge support from the Department of Atomic Energy, Government of India, under project no. RTI4001. YP acknowledges travel support from Microsoft Research for presenting the conference version of this paper. SB gratefully

acknowledges the support of the Walmart Centre for Tech Excellence (CSR WMGT-23-0001) and an Ittiam CSR Grant (OD/OTHR-24-0032).

References

A. D. Ariel D Procaccia and J. Wang. 2017. "A lower bound for equitable cake cutting." In: *Proceedings of the 2017 ACM Conference on Economics and Computation*, 479–495.

Y. Aumann and Y. Dombb. 2015. "The efficiency of fair division with connected pieces." *ACM Transactions on Economics and Computation (TEAC)*, 3, 4, 1–16.

H. Aziz and S. Rey. 2020. "Almost Group Envy-free Allocation of Indivisible Goods and Chores." In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*. Ed. by C. Bessiere. ijcai.org, 39–45.

S. Barman, V. V. Narayan, and P. Verma. 2023. "Fair Chore Division under Binary Supermodular Costs." In: *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*. Ed. by N. Agmon, B. An, A. Ricci, and W. Yeoh. ACM, 2863–2865.

U. Bhaskar, N. Misra, A. Sethia, and R. Vaish. 2023. "The Price of Equity with Binary Valuations and Few Agent Types." In: *International Symposium on Algorithmic Game Theory*. Springer, 271–289.

U. Bhaskar, A. R. Sricharan, and R. Vaish. 2025. "Connected equitable cake division via Sperner's lemma." *Inf. Process. Lett.*, 189, 106554.

I. Caragiannis, C. Kaklamanis, P. Kanellopoulos, and M. Kyropoulou. 2012. "The Efficiency of Fair Division." *Theory of Computing Systems*, 50, 4, 589–610.

K. Cechlárová, J. Doboš, and E. Pillárová. 2013. "On the existence of equitable cake divisions." *Information Sciences*, 228, 239–245.

K. Cechlárová and E. Pillárová. 2012. "On the computability of equitable divisions." *Discrete Optimization*, 9, 4, 249–257.

X. Chen and Z. Liu. 2020. "The Fairness of Leximin in Allocation of Indivisible Chores." *CoRR*, abs/2005.04864.

G. Chèze. 2017. "Existence of a simple and equitable fair division: A short proof." *Mathematical Social Sciences*, 87, 92–93.

L. E. Dubins and E. H. Spanier. 1961. "How to cut a cake fairly." *The American Mathematical Monthly*, 68, 1P1, 1–17.

D. Foley. 1966. *Resource allocation and the public sector*. Yale University.

R. Freeman, S. Sikdar, R. Vaish, and L. Xia. 2020. "Equitable Allocations of Indivisible Chores." In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 384–392.

R. Freeman, S. Sikdar, R. Vaish, and L. Xia. 2019. "Equitable Allocations of Indivisible Goods." In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 280–286.

Y. Gal, M. Mash, A. D. Procaccia, and Y. Zick. 2017. "Which Is the Fairest (Rent Division) of Them All?" *J. ACM*, 64, 6, 39:1–39:22.

P. W. Goldberg, K. Høgh, and A. Hollender. 2023. "The Frontier of Intractability for EFX with Two Agents." *International Symposium on Algorithmic Game Theory*, 14238, 290–307.

J. Goldman and A. D. Procaccia. 2015. "Spliddit: Unleashing Fair Division Algorithms." *ACM SIGecom Exchanges*, 13, 2, 41–46.

L. Gourvès, J. Monnot, and L. Tlilane. 2014. "Near Fairness in Matroids." In: *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)* (Frontiers in Artificial Intelligence and Applications). Ed. by T. Schaub, G. Friedrich, and B. O'Sullivan. Vol. 263. IOS Press, 393–398.

D. K. Herreiner and C. D. Puppe. 2009. "Envy freeness in experimental fair division problems." *Theory and decision*, 67, 65–100.

D. K. Herreiner and C. D. Puppe. 2010. "Inequality aversion and efficiency with ordinal and cardinal social preferences—An experimental study." *Journal of Economic Behavior & Organization*, 76, 2, 238–253.

J. Kagan. 2021. *Equitable Distribution: Definition, State Laws, Exempt Property*. https://www.investopedia.com/terms/e/equitable-division.asp. Accessed: 2023-08-12. (2021).

H. Moulin. 2004. *Fair division and collective welfare*. MIT press.

B. Plaut and T. Roughgarden. 2020. "Almost Envy-Freeness with General Valuations." *SIAM Journal on Discrete Mathematics*, 34, 2, 1039–1068.

A. Sun, B. Chen, and X. V. Doan. 2023. "Equitability and welfare maximization for allocating indivisible items." *Autonomous Agents and Multi-Agent Systems*, 37, 1, 8.

H. R. Varian. 1974. "Equity, Envy, and Efficiency." *Journal of Economic Theory*, 9, 1, 63–91.

A Missing Proofs from Section 3.2

In this section, we first state and prove Claim 15. The claim is then used to establish Theorem 5.

Claim 15. *After every outer iteration, either (i) p is no longer the poorest agent, and the value of agent p has increased multiplicatively by at least a factor $\left(\frac{1}{1-\varepsilon}\right)$, or (ii) all remaining unassigned goods are assigned to agent p and the algorithm terminates.*

Proof. At the start of the outer iteration, $v_p(A_p) \leq v_{p'}(A_{p'})$. The Add phase terminates only when either all remaining goods are assigned to agent p, or $v_p(A_p) > \left(\frac{1}{1-\varepsilon}\right) v_{p'}(A_{p'})$. Hence, at the end of the Add phase, either $v_p(A_p) > \left(\frac{1}{1-\varepsilon}\right) v_{p'}(A_{p'})$, or all remaining goods are assigned to agent p and $v_p(A_p) \leq \left(\frac{1}{1-\varepsilon}\right) v_{p'}(A_{p'})$. In the latter case, the Fix phase is not executed, and the algorithm terminates as claimed. In the former case, the Fix phase may be executed. Note, however, that a good $\widehat{g}$ is removed from agent p only if $v_p\left(A_p \setminus \{\widehat{g}\}\right) > \left(\frac{1}{1-\varepsilon}\right) v_{p'}(A_{p'})$. Hence, after termination of the Fix phase, we must have that $v_p(A_p) > \left(\frac{1}{1-\varepsilon}\right) v_{p'}(A_{p'})$. At this point, the value of agent p has strictly increased to above that of agent p', and by a factor of $\left(\frac{1}{1-\varepsilon}\right)$, as claimed. □

Theorem 5. *Given parameter $\varepsilon \in (0,1)$ and any fair division instance with monotone valuations, a $(1-\varepsilon)$-EQX allocation can be computed in $O\left(\frac{m^2 n}{\varepsilon} \log V_{\max}\right)$ time.*

Proof. For the time complexity, note that, from Claim 15, the value of an agent i can increase at most $\log_{\left(\frac{1}{1-\varepsilon}\right)} v_i(M) \leq \frac{\log V_{\max}}{\varepsilon}$ times. Hence, the number of outer iterations is at most $O(\ (n \log V_{\max})/\varepsilon)$. As noted in the proof of Theorem 2, the runtime of each outer iteration is $O(m^2)$. The algorithm thus terminates in $O\left(\frac{m^2 n}{\varepsilon} \log V_{\max}\right)$ time.

To prove that the allocation computed by the algorithm is indeed $(1-\varepsilon)$-EQX, we proceed as before by induction on the number of outer iterations. Initially, the allocation is empty, which is trivially EQX (and, hence, $(1-\varepsilon)$-EQX). Since the bundles assigned to agents, other than p, remain unchanged in an outer iteration, any $(1-\varepsilon)$-EQX violation must involve agent p. Let $\mathcal{B}$ be the allocation obtained after an outer iteration. Then $v_i(B_i) = v_i(A_i)$ for $i \neq p$. Also, Claim 15 ensures that $v_p(B_p) \geq v_p(A_p)$.

To show that allocation $\mathcal{B} = (B_1, \ldots, B_n)$ is $(1-\varepsilon)$-EQX, we need to show that for any agent $i \in N$, the removal of any good $g \in B_i$ reduces the value of the bundle to at most $\frac{1}{1-\varepsilon}$ times the poorest agent. This condition holds—via the induction hypothesis—for all agents $i \neq p$; recall that $B_i = A_i$, for all $i \neq p$, and $v_p(B_p) \geq v_p(A_p)$.

For agent p, note that after the completion of the Fix phase, the removal of any good from p's bundle reduces her value to at most $\left(\frac{1}{1-\varepsilon}\right) v_{p'}(A_{p'}) = \left(\frac{1}{1-\varepsilon}\right) v_{p'}(B_{p'})$. Since p was the poorest and p' was the second poorest agent in allocation $\mathcal{A}$, this implies that the removal of any good from B_p brings down p's value to below $\frac{1}{1-\varepsilon}$ times the minimum: $v_p(B_p \setminus \{g\}) \leq \left(\frac{1}{1-\varepsilon}\right) v_j(B_j)$ for all agents j and each good $g \in B_p$.

The theorem stands proved. □

B Limitation of Leximin++ under Nonmonotone Valuations

This section shows that, even in the presence of a single, non-identically valued chore, leximin++ allocations are not guaranteed to be EQX. Recall that, by contrast and for identical chores, a leximin++ allocation is always EQX (Theorem 9).

The following instance highlights the limitation of the leximin++ criterion under nonmonotone valuations. In particular, Table 5 that details the additive, objective valuations of two agents for three items: two goods, g_1, g_2 and a chore c.

Table 5. An instance where the leximin++ allocation is not EQX.

	g_1	g_2	c
Agent 1	10	1	−1
Agent 2	1	100	−1000

In the given instance, consider bundles $A_1 = \{g_1, c\}$ and $A_2 = \{g_2\}$. Under allocation $\mathcal{A} = (A_1, A_2)$, we have $v_1(A_1) = 9$ and $v_2(A_2) = 100$. Furthermore, in any other allocation, the minimum value among the two agents (i.e., the egalitarian welfare) is non-positive. Hence, for the instance at hand, $\mathcal{A}$ is the only leximin++ allocation.

However, a chores violation exists under $\mathcal{A}$, since $v_2(A_1 \setminus \{c\}) = 10 < v_2(A_2)$. Therefore, allocation $\mathcal{A}$ is not EQX.

C Extensions

In this section, we show that all our positive results hold even if we replace goods with chores and vice versa. We skip formal proofs for these results, since this would require largely repeating Sections 3 and 4. Instead, we outline the main differences encountered while establishing the converse results.

C.1 Monotone Nonincreasing Valuations

The results from Section 3 hold if the agents have *monotone nonincreasing* valuations, instead of monotone nondecreasing as earlier. That is, our results hold if all items are chores, instead of goods. In this case, Algorithm 1 requires that instead of the poorest and second-poorest agents p and p', we consider the richest and second-richest agents r and r'. The richest agent r then selects the unallocated chore with the smallest marginal value, i.e., the item which decreases her value by the largest amount. The Add phase adds chores in this manner to agent r as long as it remains a richest agent. The Fix phase removes chores from A_r that, upon removal, do not make it a richest agent. With these changes, the algorithm computes an EQX allocation in pseudo-polynomial time. This is formally stated as follows.

Theorem 16. *Given any fair division instance with monotone nonincreasing valuations, the modifications to Algorithm 1 described above compute an EQX allocation in pseudo-polynomial time.*

For polynomial time computation, instead of weakly well-layered valuations, we require that the agents' valuations be *negatively weakly well-layered*:

Definition 2. *A valuation function $v : 2^M \to \mathbb{Z}_{\leq 0}$ is said to be* negatively weakly well-layered *if for any set $M' \subseteq M$ the sets $S_0, S_1, \ldots$ obtained by the greedy algorithm (that is, $S_0 = \emptyset$ and $S_i = S_{i-1} \cup \{x_i\}$, where $x_i \in \arg\min_{x \in M' \setminus S_{i-1}} v(S_{i-1} \cup x)$, for $i \leq |M'|$) are optimal, in the sense that $v(S_i) = \min_{S \subseteq M' : |S| = i} v(S)$ for all $i \leq |M'|$.*

With these valuations, one can show that the modified Algorithm 1 runs in polynomial time, as formally stated in the following theorem.

Theorem 17. *Given any fair division instance in which all the agents have negative weakly well-layered valuations, the modifications to Algorithm 1 described above compute an EQX allocation in polynomial time.*

Lastly, for monotone nonincreasing valuations, given parameter $\varepsilon \in (0, 1)$, an allocation $\mathcal{A}$ is said to be an $(1+\varepsilon)$-EQX allocation if for every pair of agents $i, j \in N$ and for each chore $c \in A_i$ we have $v_i(A_i \setminus \{c\}) \geq (1+\varepsilon) v_j(A_j)$. Hence, in an $(1+\varepsilon)$-EQX allocation, removing any chore from any agent i's bundle improves i's value to at least $(1+\varepsilon)$ times the maximum (recall that all values are nonpositive).

We now modify Algorithm 1 as follows:

4: **while** $v_r(A_r) \geq (1+\varepsilon) v_{r'}(A_{r'})$ **and** $U \neq \emptyset$ **do**...

8: **while** there exists $\widehat{c} \in A_r$ such that $v_r(A_r \setminus \{\widehat{c}\}) < (1+\varepsilon) v_{r'}(A_{r'})$ **do**...

One can prove that this modified algorithm returns a $(1+\varepsilon)$-EQX allocation in $O\left(\frac{m^2 n}{\varepsilon} \log |V_{\min}|\right)$ time, where $V_{\min} := \min_{i \in N} v_i(M)$.

Theorem 18. *Given parameter $\varepsilon \in (0, 1)$ and any fair division instance with monotone nonincreasing valuations, a $(1+\varepsilon)$-EQX allocation can be computed in $O\left(\frac{m^2 n}{\varepsilon} \log |V_{\min}|\right)$ time.*

C.2 Additive Nonmonotone Valuations

Identically-valued goods. We show that the positive results from Section 4.2 can be obtained if we have identically-valued goods instead. In this case, agents have additive, objective valuations, and they value the goods g identically, i.e., $v_i(g) = v_j(g)$ for all agents $i, j \in N$.

In the earlier case when agents valued *chores* identically, for an allocation $\mathcal{X} = (X_1, \ldots, X_n)$ we ordered agents by increasing value $v_i(X_i)$. In this case, when agents value *goods* identically, we order agents by *increasing negation of their values* $-v_i(X_i)$ (hence, the agent with largest value appears first in this order). Thus the permutation $\sigma^{\mathcal{X}}$ is defined as:

(i) Agents with lower negated values, $-v_i(X_i)$, receive lower indices.
(ii) Among agents with equal values, agents i with lower bundle sizes $|X_i|$ receive lower indices.
(iii) Agents with equal values and number of items are ordered by the index i.

Then for allocations $\mathcal{A}$, $\mathcal{B}$, we say that $\mathcal{A} \preceq_{++} \mathcal{B}$ if, for the first index ℓ where they differ, if $i = \sigma^{\mathcal{A}}(\ell)$ and $j = \sigma^{\mathcal{B}}(\ell)$,

(i) either $-v_i(A_i) < -v_j(B_j)$,
(ii) or $-v_i(A_i) = -v_j(B_j)$ and $|A_i| < |B_j|$,
(iii) or $-v_i(A_i) = -v_j(B_j)$, $|A_i| = |B_j|$, and $i < j$.

Thus, an allocation $\mathcal{A}$ is a leximin++ allocation if $\mathcal{B} \preceq_{++} \mathcal{A}$ for all allocations $\mathcal{B}$. By Theorem 8, we know that leximin++ allocation is guaranteed to exist.

As in the earlier case, we can complete the proof for the current setting by using the observation that if a leximin++ allocation $\mathcal{A}$ is not EQX then resolving the violation gives a lexicographic improvement. The result is stated as follows.

Theorem 19. *In a fair division instance, if the agents have additive, objective valuations and they value the goods identically, then the leximin++ allocation is EQX.*

Additive chores and a single good. Algorithm 5 is modified in the natural manner, replacing goods with chores and vice versa.

Algorithm 8 Algorithm for single good setting

Input: Fair division instance $(N, M, \mathcal{V})$ with additive, objective valuations and a single good g.
Output: EQX allocation $\mathcal{A}$

Initialize $A_1 = M$ and $A_i = \emptyset$ for all agents $i \neq 1$.
while $\mathcal{A} = (A_1, \ldots, A_n)$ is **not** EQX **do**
  $r = \sigma^{\mathcal{A}}(1)$. {$r$ is the first agent according to $\sigma^{\mathcal{A}}$.}
  while there exists agent $i \in N$ and chore $c \in A_i$ such that $v_i(A_i \setminus \{c\}) < v_r(A_r)$ **do**
    Update $A_r \leftarrow A_r \cup \{c\}$ and $A_i \leftarrow A_i \setminus \{c\}$.
    Update $r = \sigma^{\mathcal{A}}(1)$.
  end while {After resolving all chores violations, check for goods violation.}
  $p \leftarrow \sigma^{\mathcal{A}}(n)$. {$p$ is the last agent according to $\sigma^{\mathcal{A}}$.}
  if $v^+(\mathcal{A}) > v_p(A_p)$ **then**
    Update $A_{\kappa(\mathcal{A})} \leftarrow A_{\kappa(\mathcal{A})} \setminus \{g\}$ and $A_p \leftarrow A_p \cup \{g\}$.
  end if
end while
return $\mathcal{A}$

Theorem 20. *Given any fair division instance with additive objective valuations and a single good, Algorithm 8 terminates in finite time and returns an EQX allocation.*

Proof sketch. As before, in an allocation $\mathcal{A}$, $\kappa(\mathcal{A})$ is the agent holding the unique *good* g, and the value $v^+(\mathcal{A}) := v_{\kappa(\mathcal{A})}(A_{\kappa(\mathcal{A})} \setminus g)$ is the *cutoff value*. In case of a goods violation we have $v_{\kappa(\mathcal{A})}(A_{\kappa(\mathcal{A})} \setminus g) > v_p(A_p)$. Let $\mathcal{B}$ denote the allocation after transferring good g to agent p. Note that such a good transfer decreases the cutoff value: under allocation $\mathcal{A}$, the cutoff value is $v^+(\mathcal{A}) = v_{\kappa(\mathcal{A})}(A_{\kappa(\mathcal{A})} \setminus g)$ and, under the updated allocation $\mathcal{B}$, it is $v^+(\mathcal{B}) = v_p(A_p)$. In particular, $v^+(\mathcal{A}) > v^+(\mathcal{B})$. As before, in our analysis, we will keep track of two progress measures separately: the lexicographic value of the allocation (according to the order $\preceq_{++}$ defined for identically-valued goods and the cutoff value of the allocation. We can show that the cutoff value of the allocation is nonincreasing between successive goods violations, and strictly decreases whenever a goods violation is resolved. Whenever a chores violation is resolved, we obtain a lexicographic improvement in the allocation; a goods violation may however result in a lexicographic decrease. Since the cutoff value can decrease and the lexicographic value can increase only a finite number of times, the local search algorithm must terminate in finite time. □

Received 8 June 2024; accepted 14 September 2025