The Good, the Bad and the Submodular: Fairly Allocating Mixed Manna Under Order-Neutral Submodular Preferences

Cyrus Cousins[0000−0002−1691−0282], Vignesh Viswanathan[0000−0002−6261−6228], and Yair Zick[0000−0002−0635−6230]

University of Massachusetts, Amherst, MA 01002, USA
`{cbcousins, vviswanathan, yzick}@umass.edu`

Abstract. We study the problem of fairly allocating indivisible goods (positively valued items) and chores (negatively valued items) among agents with decreasing marginal utilities over items. Our focus is on instances where all the agents have *simple* preferences; specifically, we assume the marginal value of an item can be either -1, 0 or some positive integer c. Under this assumption, we present an efficient algorithm to compute leximin allocations for a broad class of valuation functions we call *order-neutral* submodular valuations. Order-neutral submodular valuations strictly contain the well-studied class of additive valuations but are a strict subset of the class of submodular valuations. We show that these leximin allocations are Lorenz dominating and approximately proportional. We also show that, under further restriction to additive valuations, these leximin allocations are approximately envy-free and guarantee each agent their maximin share. We complement this algorithmic result with a lower bound showing that the problem of computing leximin allocations is NP-hard when c is a rational number.

Keywords: Fair Allocation · Indivisible Items · Mixed Manna.

1 Introduction

Fair allocation is a fundamental problem in computational economics. The problem asks how to divide a set of indivisible items among agents with subjective preferences (or valuations) over the items. Most of the literature focuses on the problem of dividing *goods* — items with positive value. However, in several practical applications, such as dividing a set of tasks or allocating shifts to employees, items can be *chores* which provide a negative value to the agents they are allocated to.

Fair allocation with mixed manna (instances containing both goods and chores) is, unsurprisingly, a harder problem than the case with only goods. Several questions which have been answered positively in the only goods setting are either still open or face a negative (impossibility or intractability) result in the mixed goods and chores setting. In particular, very little is known about the computability of *leximin allocations*. A leximin allocation is one that maximizes the utility of the agent with the least utility; subject to that, it maximizes the second-least utility, and so on.

In the only goods setting, maximizing Nash welfare is arguably one of the most popular fairness objectives. Unfortunately, the Nash welfare of an allocation, defined as the product of agent utilities, loses its meaning in settings where agent utilities can be negative. In such settings, the leximin objective is a natural substitute. The leximin objective is easy to understand, and it implies an appealing egalitarian notion of fairness. Therefore, designing algorithms that efficiently compute exact (or approximate) leximin allocations in the mixed manna setting is an important research problem.

In this work, we present the first non-trivial results for this problem.

1.1 Our Results

Contributions to the Theory of Fair Allocation. We take a systematic approach towards solving this problem and start with a simple class of valuation functions. We assume all goods are symmetric and provide value c (where c is a positive integer), and all chores are symmetric and provide value -1. We also allow items to provide a value of 0. We assume there are decreasing marginal gains over items; that is, after receiving many items, a good may provide value 0 or even turn into a chore and provide a value of -1. We refer to this class of valuations as $\{-1, 0, c\}$-submodular valuations. With such valuations, there is no clear demarcation between goods and chores; an item may provide positive marginal value when added to an empty bundle but provide negative marginal value when added to a non-empty bundle.

We present an algorithm to compute a leximin allocation for a broad subclass of $\{-1, 0, c\}$ submodular valuations. More specifically, we show that leximin allocations can be computed efficiently when agents have $\{-1, 0, c\}$ submodular valuation functions that satisfy an additional property we call *order-neutrality.* Order-neutrality can be very loosely thought of as a property that requires the number of c-valued items in a bundle to be a monotonically non-decreasing function of the bundle. We analyze these leximin allocations in further detail showing that they are Lorenz dominating and proportional up to one item. We also show that under further restriction to the case where agents have $\{-1, 0, c\}$ additive (or linear) valuations, leximin allocations are approximately envy-free as well as offer each agent their maximin share. We complement this result with lower bounds showing that the problem of computing leximin allocations becomes computationally intractable when c is relaxed to being an arbitrary rational number as opposed to a positive integer.

Technical Contributions. We put forward several interesting combinatorial and algorithmic contributions. We introduce an interesting function class — Order-Neutral Submodular (ONSUB) Valuations. This is a subclass of submodular valuations for which items' marginal contributions takes on a regular structure. While we only analyze this class within the context of fair allocation of indivisible goods, we suspect that the function class may serve as a useful object of study in other domains featuring agents with combinatorial valuations such as matching markets, committee election and participatory budgeting.

In addition, we introduce the weighted item exchange graph. This is an extension of the exchange graph used in matroid theory [24], and more recently in the fair allocation literature [7, 4, 27, 25] to analyze the specific case where all agents have binary submodular valuations. The weighted exchange graph is a more general theoretical tool which allows us to carefully manipulate allocations when agents do not necessarily have binary submodular valuations.

Finally, our key algorithmic contribution is Algorithm 1 which computes leximin allocations. Algorithm 1 operates in three phases: first, it computes a utility maximizing partial allocation. Next, it uses transfer paths to obtain a partial leximin allocation. These partial allocations do not allocate any items that provide a negative marginal value to the agents they are allocated to. To make the allocation complete and allocate the final set of items, the algorithm proceeds to greedily allocate items, whose marginal gain is -1, to the highest utility agents.

1.2 Related Work

Fair allocation with mixed goods and chores has recently gained popularity in the literature. [2] presents definitions of envy-free up to one item (`EF1`) and proportionality up to one item (`PROP1`) for the mixed goods and chores setting; [2] also presents an algorithm to efficiently compute `EF1` allocations. [11] and [9] further study the existence and computation of approximate envy-free allocations. There have also been a couple of papers studying maximin share fairness with mixed goods and chores. [18] shows that there exists a PTAS to compute a maximin share fair allocation under certain assumptions. On the other hand, [19] shows that the problem of approximating the maximin share of each agent is computationally intractable under additive valuations.

To the best of our knowledge, there are only two works [11, 9] on the fair allocation of mixed goods and chores which present results for non-additive valuation classes. [11] presents several positive results for very specific cases, such as identical valuations, Boolean valuations and settings with two agents. [9] presents an algorithm to compute `EF1` allocations under *doubly monotone valuations*. Doubly monotone valuations assume each item is a good (always has positive marginal gain) or a chore (always has negative marginal gain) but otherwise, do not restrict the valuations. Order-neutral submodular valuations (discussed in this paper) relax the assumption that each item must be classified as a good or a chore, but come with the stronger restriction of submodularity. We also note that [2] presents an algorithm for computing an `EF1` allocation for doubly monotone valuations; however, this algorithm's correctness was disproved by [9].

The domain restrictions described above are not uncommon in fair allocation. In the mixed goods and chores setting, [17] studies the problem of fair allocation with lexicographic valuations — a restricted subclass of additive valuations. In the goods setting, binary valuations [16, 25, 27, 4, 8] and bivalued valuations [1, 14] have been extensively studied. In the chores setting, bivalued additive valuations [13, 3, 15] and binary submodular valuations [5, 26] have been well studied. Our results are a natural extension of this line of work.

2 Preliminaries

We use $[k]$ to denote the set $\{1, 2, \dots, k\}$. Given a set S and an element o, we use $S + o$ and $S - o$ to denote the sets $S \cup \{o\}$ and $S \setminus \{o\}$ respectively.

We have a set of n *agents* $N = [n]$ and a set of m *items* $O = \{o_1, o_2, \dots, o_m\}$. Each agent $i \in N$ has a *valuation function* $v_i : 2^O \to \mathbb{R}$; $v_i(S)$ denotes the value of the set of items S according to agent i. Given a valuation function v, we let $\Delta_v(S, o) = v(S + o) - v(S)$ denote the marginal utility of adding the item o to the bundle S under v. When clear from context, we sometimes write $\Delta_i(S, o)$ instead of $\Delta_{v_i}(S, o)$ to denote the marginal utility of giving the item o to agent i given that they have already been assigned the bundle S.

An *allocation* $X = (X_0, X_1, \dots, X_n)$ is an $(n+1)$-partition of the set of items O. X_i denotes the set of items allocated to agent i and X_0 denotes the set of unallocated items. Our goal is to compute *complete* fair allocations — allocations where $X_0 = \emptyset$. When we construct an allocation, we sometimes only define the allocation to each agent $i \in N$; the bundle X_0 is implicitly assumed to contain all the unallocated items. Given an allocation X, we refer to $v_i(X_i)$ as the utility of agent i under the allocation X. We also define the utility vector of an allocation X as the vector $\boldsymbol{u}^X = (v_1(X_1), v_2(X_2), \dots, v_n(X_n))$.

We define two common methods to compare vectors. We will use these methods extensively in our analysis when comparing allocations.

Definition 2.1 (Lexicographic Dominance). *A vector $\boldsymbol{y} \in \mathbb{R}^n$* lexicographically dominates *a vector $\boldsymbol{z} \in \mathbb{R}^n$ (written $\boldsymbol{y} \succ_{\texttt{lex}} \boldsymbol{z}$) if there exists a $k \in [n]$ such that for all $j \in [k-1]$, $y_j = z_j$ and $y_k > z_k$. We sometimes say an allocation X lexicographically dominates an allocation Y if $\boldsymbol{u}^X \succ_{\texttt{lex}} \boldsymbol{u}^Y$.*

Definition 2.2 (Pareto Dominance). *A vector $\boldsymbol{y} \in \mathbb{R}^n$* Pareto dominates *a vector $\boldsymbol{z} \in \mathbb{R}^n$ if for all $j \in [n]$, $y_j \geq z_j$ with the inequality being strict for at least one $j \in [n]$. An allocation X Pareto dominates an allocation Y if $\boldsymbol{u}^X$ Pareto dominates $\boldsymbol{u}^Y$.*

2.1 Valuation Functions

In this paper, we will be dealing with the popular class of submodular valuations.

Definition 2.3 (Submodular functions). *A function $v : 2^O \to \mathbb{R}$ is a* submodular function *if (a) $v(\emptyset) = 0$, and (b) for any $S \subseteq T \subseteq O$ and $o \in O \setminus T$, $\Delta_v(S, o) \geq \Delta_v(T, o)$.*

We also define restricted submodular valuations to formally capture instances where the function has a limited set of marginal values. Throughout this paper, we use the set A to denote an arbitrary set of real numbers.

Definition 2.4 (A-SUB functions). *Given a set of real numbers A, a function $v : 2^O \to \mathbb{R}$ is an A-SUB function if it is submodular, and every item's marginal contribution is in A. That is, for any set $S \subseteq O$ and an item $o \in O \setminus S$, $\Delta_v(S, o) \in A$.*

Our analysis of submodular functions requires that they satisfy an additional property we call *order-neutrality.* Given a submodular valuation function $v : 2^O \to \mathbb{R}$, the value of a set of items S can be computed by adding items from S one by one into an empty set and adding up the $|S|$ marginal gains. More formally, given a bijective mapping $\pi : [|S|] \to S$ which defines the order in which items are added, $v(S)$ can be written as the following telescoping sum:

$$v(S) = \sum_{j \in [|S|]} \Delta_v \left(\bigcup\nolimits_{\ell \in [j-1]} \pi(\ell), \pi(j) \right).$$

While the value $v(S)$ does not depend on π, the values of the marginal gains in the telescoping sum may depend on π. Given a set S and a bijective mapping $\pi : [|S|] \to S$ we define the vector $\boldsymbol{v}(S, \pi)$ as the vector of marginal gains (given below) *sorted in ascending order*

$$\Big(\Delta_v\big(\emptyset, \pi(1)\big), \Delta_v\big(\pi(1), \pi(2)\big), \ldots, \Delta_v\big(S - \pi(|S|), \pi(|S|)\big) \Big).$$

We refer to this vector $\boldsymbol{v}(S, \pi)$ as a *sorted telescoping sum vector* . For any ordering π and set S, the sum of the elements of $\boldsymbol{v}(S, \pi)$ is equal to $v(S)$. A submodular function v is said to be *order-neutral* if for all bundles $S \subseteq O$ and any two orderings $\pi, \pi' : [|S|] \to S$ of items in the bundle S, we have $\boldsymbol{v}(S, \pi) = \boldsymbol{v}(S, \pi')$; that is, any sorted telescoping sum vector is independent of the order π. For order-neutral submodular valuations, we sometimes drop the π and refer to any sorted telescoping sum vector using $\boldsymbol{v}(S)$. This definition can be similarly extended to A-SUB functions. For readability, we refer to order-neutral A-SUB functions as A-ONSUB functions.

Definition 2.5 (A-ONSUB functions). *Given a set of real numbers A, a function $v : 2^O \to \mathbb{R}$ is an A-ONSUB function if it is both order-neutral and an A-SUB function.*

We will focus on instances with $\{-1, 0, c\}$-ONSUB valuations where c is a positive integer. Throughout this paper, the only use of c will be to denote an arbitrary positive integer. We also present some results for the restricted setting where agents have additive valuations.

Definition 2.6 (A-ADD functions). *Given a set of real numbers A, a function $v : 2^O \to \mathbb{R}$ is an A-additive (or simply A-ADD) function if $v(\{o\}) \in A$ for all $o \in O$ and $v(S) = \sum_{o \in S} v(\{o\})$ for all $S \subseteq O$.*

To build intuition for the class of $\{-1, 0, c\}$-ONSUB valuations, we present a few simple examples below.

Example 2.7. Let $O = \{o_1, o_2, o_3, o_4\}$, the following functions $v_1, v_2, v_3 : 2^O \to \mathbb{R}$ are $\{-1, 0, c\}$-ONSUB:

$$\begin{aligned} v_1(S) &= c \min\{|S|, 2\}, \\ v_2(S) &= c\mathbb{I}\{o_1 \in S\} - \mathbb{I}\{o_2 \in S\}, \\ v_3(S) &= c \min\{|S \cap \{o_1, o_2\}|, 1\} - |S \cap \{o_3, o_4\}|. \end{aligned}$$

Agent 1's valuation v_1 describes a function where any item provides a value of c but the marginal utility of any item drops to 0 after two items are added to the bundle. v_2 describes a simple additive function where o_1 provides a value of c and o_2 provides a value of -1. v_3 describes a slightly more complex function where o_1 and o_2 are goods but at most one of them can a provide a marginal value of c; o_3 and o_4 are chores providing a marginal value of -1 each.

Note that additive valuations are trivially order-neutral submodular valuations. Unsurprisingly, not all submodular valuations are order-neutral — consider a function v over two items $\{o_1, o_2\}$ such that $v(\{o_1\}) = 0$, $v(\{o_2\}) = 1$ and $v(\{o_1, o_2\}) = 0$. This function is submodular, but not order-neutral, since $v(\{o_1, o_2\})$ has two different sorted telescoping sum vectors. However, it is worth noting that there are many interesting non-additive order-neutral submodular functions. For example, any binary submodular function ($\{0, 1\}$-SUB) is order-neutral.

Proposition 2.8. *When $|A| = 2$, any A-SUB function is order-neutral.*

This proposition implies that the class of $\{0, c\}$-SUB, $\{-1, c\}$-SUB and $\{-1, 0\}$-SUB valuations are all contained in the class of $\{-1, 0, c\}$ order-neutral submodular valuations. It is also worth noting that capped additive valuations, where agents can only receive a positive marginal utility from a fixed number of items also falls under the class of $\{-1, 0, c\}$ order-neutral submodular valuations.

2.2 Fairness and Efficiency Objectives

There are several reasonable fairness objectives used in the fair allocation literature. We discuss most of them in this paper. However, to avoid an overload of definitions, we only define the following two fairness and efficiency objectives in this section.

Utilitarian Social Welfare (`USW`): The utilitarian social welfare of an allocation X is $\sum_{i \in N} v_i(X_i)$. An allocation X is said to be `MAX-USW` if it maximizes the utilitarian social welfare.

Leximin: An allocation is said to be leximin if it maximizes the utility of the least valued agent, and subject to that, the utility of the second-least valued agent, and so on [20]. This is usually formalized using the sorted utility vector. The *sorted utility vector* of an allocation X (denoted by $\boldsymbol{s}^X$) is defined as the utility vector $\boldsymbol{u}^X$ sorted in ascending order. An allocation X is leximin if there is no allocation Y such that $\boldsymbol{s}^Y \succ_{\texttt{lex}} \boldsymbol{s}^X$.

Other objectives like maximin share, proportionality and envy-freeness are defined in Section 6.

2.3 Exchange Graphs and Path Augmentations

In this section, we describe the classic technique of path augmentations. A modified version of these path augmentations is used extensively in our algorithm design. Path augmentations have been used to carefully manipulate allocations when agents have binary submodular valuations ($\{0,1\}$-SUB functions). However, they only work with *clean* allocations.

Definition 2.9 (Clean Allocation). *For any agent $i \in N$, a bundle S is* clean *with respect to the binary submodualar valuation β_i if $\beta_i(S) = |S|$. An allocation X is said to be clean (w.r.t .$\{\beta_h\}_{h\in N}$) if for all agents $i \in N$, $\beta_i(X_i) = |X_i|$.*

When agents have binary submodular valuations $\{\beta_h\}_{h\in N}$, given a clean allocation X (w.r.t. $\{\beta_h\}_{h\in N}$), we define the *exchange graph* $\mathcal{G}(X,\beta)$ as a directed graph over the set of items O, where an edge exists from o to o' in the exchange graph if $o \in X_j$ and $\beta_j(X_j - o + o') = \beta_j(X_j)$ for some $j \in N$. In other words, an edge exists if agent j can swap item o with the item o' and still retain the same utility level. There is no outgoing edge from any item in X_0.

Let $P = (o_1, o_2, \dots, o_t)$ be a path in the exchange graph $\mathcal{G}(X,\beta)$ for a clean allocation X. We define a transfer of items along the path P in the allocation X as the operation where o_t is given to the agent who has o_{t-1}, o_{t-1} is given to the agent who has o_{t-2}, and so on until finally o_1 is discarded and becomes freely available. This transfer is called *path augmentation*; the bundle X_i after path augmentation with the path P is denoted by $X_i \Lambda P$ and defined as $X_i \Lambda P = (X_i - o_t) \oplus \{o_j, o_{j+1} : o_j \in X_i\}$, where $\oplus$ denotes the symmetric set difference operation.

For any clean allocation X and agent i, we define $F_{\beta_i}(X,i) = \{o \in O : \Delta_{\beta_i}(X_i, o) = 1\}$ as the set of items which give agent i a marginal gain of 1 under the valuation β_i. For any agent i, let $P = (o_1, \dots, o_t)$ be a *shortest* path from $F_{\beta_i}(X,i)$ to X_j for some $j \neq i$. Then path augmentation with the path P and giving o_1 to i results in a clean allocation where the size of i's bundle $|X_i|$ goes up by 1, the size of j's bundle goes down by 1 and all the other agents do not see any change in size. This is formalized below and exists in [6, Lemma 1] and [27, Lemma 3.7].

Lemma 2.10 ([6], [27]). *Let X be a clean allocation with respect to the binary submodular valuations $\{\beta_h\}_{h\in N}$. Let $P = (o_1, \dots, o_t)$ be a shortest path in the exchange graph $\mathcal{G}(X,\beta)$ from $F_{\beta_i}(X,i)$ to X_j for some $i \in N$ and $j \in N + 0 - i$. Then, the following allocation Y is clean with respect to $\{\beta_h\}_{h\in N}$.*

$$Y_k = \begin{cases} X_k \Lambda P & (k \in N + 0 - i) \\ X_i \Lambda P + o_1 & (k = i) \end{cases}$$

Moreover, for all $k \in N + 0 - i - j$, $|Y_k| = |X_k|$, $|Y_i| = |X_i| + 1$ and $|Y_j| = |X_j| - 1$.

We also present sufficient conditions for a path to exist. A slight variant of the following lemma appears in [27, Theorem 3.8].

Lemma 2.11 ([27]). *Let X and Y be two clean allocations with respect to the binary submodular valuations $\{\beta_h\}_{h\in N}$. For any agent $i \in N$ such that $|X_i| < |Y_i|$, there is a path from $F_{\beta_i}(X, i)$ to either*

(i) some item in X_k for some $k \in N$ in the exchange graph $\mathcal{G}(X, \beta)$ such that $|X_k| > |Y_k|$, or

(ii) some item in X_0 in $\mathcal{G}(X, \beta)$.

This technique of path augmentations has been extensively exploited in the design of the Yankee Swap algorithm [27]. Given an instance with binary submodular valuations, Yankee Swap computes a clean `MAX-USW` leximin allocation in polynomial time. We use this procedure as a subroutine in our algorithm to compute a partial allocation.

Theorem 2.12 (Yankee Swap [27]). *When agents have binary submodular valuations $\{\beta_h\}_{h\in N}$, there exists an efficient algorithm that computes a clean `MAX-USW` leximin allocation.*

We note that Yankee Swap is not the only algorithm to compute clean leximin allocations; [4] also presents an efficient algorithm to do so.

3 Understanding A-ONSUB Valuations

We now present a result about A-ONSUB valuations exploring its connection to binary submodular functions. This connection allows us to adapt path augmentations for our setting. Our arguments generalize the arguments presented by [12, Section 3] about bivalued submodular valuations.

Our result shows that given a threshold value $\tau \in \mathbb{R}$ and an A-ONSUB function v_i, the number of values in the sorted telescoping sum vector greater than or equal to the threshold τ corresponds to a binary submodular function. More formally, for any bundle $S \subseteq O$ and agent $i \in N$, let $\beta_i^\tau(S)$ denote the number of values in the sorted telescoping sum vector $\boldsymbol{v}_i(S)$ greater than or equal to τ. We show that the function β_i^τ is a binary submodular function.

Lemma 3.1. *For any $i \in N$, the function β_i^τ is a binary submodular function.*

This lemma is particularly useful: if any allocation is clean with respect to the valuations $\{\beta_h^\tau\}_{h\in N}$, we can use path augmentations to modify the allocation. In our analysis, we will extensively use the valuations $\{\beta_h^0\}_{h\in N}$ and $\{\beta_h^c\}_{h\in N}$ corresponding to the cases where $\tau = 0$ and $\tau = c$ respectively.

4 Understanding $\{-1, 0, c\}$-ONSUB Valuations

We turn our attention to the specific set of valuations assumed in this paper — $\{-1, 0, c\}$-ONSUB valuations for some positive integer c. We establish a few important technical lemmas for fair allocation instances when all agents have $\{-1, 0, c\}$-ONSUB valuations.

We first show that any allocation X can be decomposed into *three* allocations X^{-1}, X^0 and X^c such that the items with marginal value c are in X^c, items with marginal value 0 are in X^0 and items with marginal value -1 are in X^{-1}.

Lemma 4.1. *When agents have $\{-1, 0, c\}$-ONSUB valuations, for any allocation X, there exist three allocations X^{-1}, X^0, and X^c such that for each agent $i \in N$ (a) $X_i^{-1} \cup X_i^0 \cup X_i^c = X_i$, (b) X_i^{-1}, X_i^0, and X_i^c are pairwise disjoint, (c) $v_i(X_i^c \cup X_i^0) = v_i(X_i^c) = c|X_i^c|$, and (d) $v_i(X_i) = c|X_i^c| - |X_i^{-1}|$.*

We use $Y = Y^c \cup Y^0 \cup Y^{-1}$ to denote the decomposition of any allocation Y into three allocations satisfying the conditions of Lemma 4.1. More generally, given any two allocations X and Y, we refer to the allocation $X \cup Y$ as the allocation where each agent $i \in N$ receives the bundle $X_i \cup Y_i$.

Example 4.2. Consider an instance with two agents $\{1, 2\}$ and four items $\{o_1, o_2, o_3, o_4\}$. Agent valuations are defined as follows:

$$v_1(S) = c \min\{|S \cap \{o_1, o_2, \}|, 1\}, \qquad v_2(S) = -|S \cap \{o_3, o_4\}|.$$

Consider an allocation X where $X_1 = \{o_1, o_2\}$ and $X_2 = \{o_3, o_4\}$. The following is a valid decomposition of X:

$$\begin{aligned} X_0^c &= \{o_2, o_3, o_4\} & X_0^0 &= \{o_1, o_3, o_4\} & X_0^{-1} &= \{o_1, o_2\} \\ X_1^c &= \{o_1\} & X_1^0 &= \{o_2\} & X_1^{-1} &= \emptyset \\ X_2^c &= \emptyset & X_2^0 &= \emptyset & X_1^{-1} &= \{o_3, o_4\} \end{aligned}$$

There may be other decompositions as well. Specifically, if we swap o_1 and o_2 in the above decomposition, the new set of allocations still corresponds to a valid decomposition.

Note that while order-neutral submodular valuations can have multiple decompositions, $\{-1, 0, c\}$-ADD valuations have a unique decomposition — for any allocation $X = X^c \cup X^0 \cup X^{-1}$ and agent i, X_i^c consists of all items in X_i that agent i values at c, X_i^0 consists of the items that i values at 0 and X_i^{-1} consists of the items that i values at -1.

4.1 Weighted Exchange Graphs

Given an allocation $X = X^c \cup X^0 \cup X^{-1}$, the path augmentation technique introduced in Section 2 can be used to manipulate X^c (using the exchange graph $\mathcal{G}(X^c, \beta^c)$) or $X^c \cup X^0$ (using the exchange graph $\mathcal{G}(X^c \cup X^0, \beta^0)$). However, when we use path augmentation with the exchange graph $\mathcal{G}(X^c, \beta^c)$ to manipulate the allocation X^c, we may affect the cleanness of $X^c \cup X^0$ (w.r.t. the valuations $\{\beta_h^0\}_{h \in N}$). To see why, consider the following simple example.

Example 4.3. Consider an example with one agent $\{1\}$ and two items $\{o_1, o_2\}$. The agent's valuation function is defined as follows:

$$v_1(S) = c \min\{|S|, 1\} - \max\{|S| - 1, 0\}.$$

In simple words, the first item in the bundle gives agent 1 a value of c but the second item gives agent 1 a marginal value of -1. Consider the allocations X^c and X^0, where $X_1^c = \emptyset$ and $X_1^0 = \{o_1\}$.

Note that X^c is clean with respect to $\{\beta_h^c\}_{h \in N}$ and $X^c \cup X^0$ is clean with respect to $\{\beta_h^0\}_{h \in N}$. The singleton path (o_2) is one of the shortest paths from $F_{\beta_1^c}(X^c)$ to X_0^c in the exchange graph $\mathcal{G}(X^c, \beta^c)$. Augmenting along this path creates an allocation Y where $Y_1^c = \{o_2\}$ and $X_1^0 = \{o_1\}$.

Validating the correctness of Lemma 2.10, Y^c is indeed clean with respect to $\{\beta_h^c\}_{h \in N}$. However, $Y^c \cup X^0$ is not clean with respect to $\{\beta_h^0\}_{h \in N}$. Note that if we instead chose to augment along the singleton path (o_1) instead of (o_2), we would have not faced this issue.

In the above example, note that X^c and X^0 do not form a valid decomposition of X. This is deliberate; in our algorithm design, we will not assume that X^c, X^0 and X^{-1} form a valid decomposition of X. We will only assume that X^c is clean with respect to $\{\beta_h^c\}_{h \in N}$ and $X^c \cup X^0$ is clean with respect to $\{\beta_h^0\}_{h \in N}$. Our goal is to use path augmentations to modify X^c while retaining both these useful properties. To guarantee that $X^c \cup X^0$ remains clean (w.r.t. $\{\beta_h^0\}_{h \in N}$) even after path augmentation, we present a technique to carefully choose paths in the exchange graph. On a high level, this is done by giving weights to the edges in the exchange graph and choosing the least-weight path as opposed to a shortest path. The way we weigh edges is motivated by Example 4.3 — for all $i \in N$, we give edges from X_i^c to X_i^0 a lower weight than other edges. It turns out that this simple change is sufficient to ensure the cleanness of $X^c \cup X^0$ is clean with respect to $\{\beta_h^0\}_{h \in N}$.

More formally, we define the weighted exchange graph $\mathcal{G}^w(X^c, X^0, \beta^c)$ as a weighted directed graph with the same nodes and edges as $\mathcal{G}(X^c, \beta^c)$. Each edge has a specific weight defined as follows: all edges from $o \in X_i^c$ to $o' \in X_i^0$ for any $i \in N$ are given a weight of $\frac{1}{2}$; the remaining edges are given a weight of 1.

For any agent $i \in N$, we define two paths on this weighted exchange graph. A *Pareto-improving path* is a path from $F_{\beta_i^c}(X^c, i)$ for some $i \in N$ to some item in X_0^c. An *exchange path* is a path from $F_{\beta_i^c}(X^c, i)$ for some $i \in N$ to some item in X_j^c for some $j \in N - i$. We first show that path augmentation along the least-weight Pareto-improving path maintains the cleanness of $X^c \cup X^0$.

Theorem 4.4 (Pareto Improving Paths). *When agents have $\{-1, 0, c\}$-ONSUB valuations, let X^c be a clean allocation with respect to $\{\beta_h^c\}_{h \in N}$ and X^0 be an allocation such that $X^c \cup X^0$ is clean with respect to the valuations $\{\beta_h^0\}_{h \in N}$. Let $X_h^c \cap X_h^0 = \emptyset$ for all $h \in N$. For some agent $i \in N$, if a Pareto-improving path exists from $F_{\beta_i^c}(X^c, i)$ to X_0^c in the weighted exchange graph $\mathcal{G}^w(X^c, X^0, \beta^c)$, then, path augmentation along the least-weight Pareto-improving path $P = (o_1, \dots, o_t)$ from $F_{\beta_i^c}(X^c, i)$ to X_0^c in the weighted exchange graph $\mathcal{G}^w(X^c, X^0, \beta^c)$ results in the following allocations Y^c and Y^0:*

$$Y_k^c = \begin{cases} X_k^c \Lambda P & (k \in N + 0 - i) \\ X_i^c \Lambda P + o_1 & (k = i) \end{cases}, \qquad Y_k^0 = \begin{cases} X_k^0 - o_t & (k \in N) \\ X_0^0 + o_t & (k = 0) \end{cases}.$$

Y^c is clean with respect to with respect to $\{\beta^c_h\}_{h\in N}$ and $Y^c \cup Y^0$ is clean with respect to $\{\beta^0_h\}_{h\in N}$. Furthermore, for all $h \in N$, $Y^c_h \cap Y^0_h = \emptyset$.

In the above Theorem, we also modify Y^0 as part of the path augmentation operation to ensure o_t is not present in two different bundles. We also show that when Pareto-improving paths do not exist, the least-weight exchange path maintains the cleanness of $X^c \cup X^0$.

Theorem 4.5 (Exchange Paths). *When agents have $\{-1, 0, c\}$-ONSUB valuations, let X^c be a clean allocation with respect to $\{\beta^c_h\}_{h\in N}$ and X^0 be an allocation such that $X^c \cup X^0$ is clean with respect to the valuations $\{\beta^0_h\}_{h\in N}$. Let $X^c_h \cap X^0_h = \emptyset$ for all $h \in N$. If no Pareto-improving path exists from $F_{\beta^c_i}(X, i)$ to X^c_0 in the weighted exchange graph $\mathcal{G}^w(X^c, X^0, \beta^c)$, then, path augmentation along the least-weight exchange path $P = (o_1, \dots, o_t)$ from $F_{\beta^c_i}(X, i)$ to X^c_j (for any $j \in N - i$) in the weighted exchange graph $\mathcal{G}^w(X^c, X^0, \beta^c)$ results in the allocations Y^c and Y^0:*

$$Y^c_k = \begin{cases} X^c_k \Lambda P & (k \in N + 0 - i) \\ X^c_i \Lambda P + o_1 & (k = i) \end{cases}, \qquad Y^0_k = X^0_k \qquad (k \in N).$$

Y^c is clean with respect to with respect to $\{\beta^c_h\}_{h\in N}$ and $Y^c \cup Y^0$ is clean with respect to with respect to $\{\beta^0_h\}_{h\in N}$. Furthermore, for all $h \in N$, $Y^c_h \cap Y^0_h = \emptyset$.

Note that since $\mathcal{G}^w(X^c, X^0, \beta^c)$ and $\mathcal{G}(X^c, \beta^c)$ have the same set of edges, Lemma 2.11 applies to the weighted exchange graph as well.

Lemma 4.6. *When agents have $\{-1, 0, c\}$-ONSUB valuations, let X^c be a clean allocation with respect to $\{\beta^c_h\}_{h\in N}$ and X^0 be an allocation such that $X^c \cup X^0$ is clean with respect to the valuations $\{\beta^0_h\}_{h\in N}$. Let $X^c_h \cap X^0_h = \emptyset$ for all $h \in N$. For any $i \in N$ and $j \in N+0$, there is a path from $F_{\beta^c_i}(X^c, i)$ to X^c_j in $\mathcal{G}^w(X^c, X^0, \beta^c)$ if and only if there is a path from $F_{\beta^c_i}(X^c, i)$ to X^c_j in $\mathcal{G}(X^c, \beta^c)$.*

Given an allocation $X = X^c \cup X^0 \cup X^{-1}$, these results already hint at a method to modify X^c such that it becomes a leximin allocation with respect to the valuations $\{\beta^c_h\}_{h\in N}$: greedily use path augmentations in the weighted exchange graph until X^c is leximin. This is the high-level approach we use in our algorithm.

5 Leximin Allocations with $\{-1, 0, c\}$-ONSUB Valuations

We are ready to present our algorithm to compute leximin allocations. Our algorithm has three phases. In the first phase, we use Yankee Swap (Theorem 2.12) to compute a `MAX-USW` allocation X with respect to the valuations $\{\beta^0_i\}_{i\in N}$. We also initialize X^c and X^{-1} to be empty allocations.

In the second phase, we update X^c and X^0 using path augmentations (from Section 4.1) until X^c is a leximin partial allocation. This is done by greedily

ALGORITHM 1: Leximin Allocations with $\{-1,0,c\}$-ONSUB Valuations

```
Input : A set of items $O$ and a set of agents $N$ with $\{-1,0,c\}$-ONSUB
        valuations $\{v_h\}_{h \in N}$
Output: A complete leximin allocation
   // Phase 1: Make $X^c \cup X^0$ a MAX-USW allocation w.r.t. $\{\beta^0_h\}_{h \in N}$
1  $X^0 \leftarrow$ the output of Yankee Swap with respect to $\{\beta^0_h\}_{h \in N}$
2  $X^c = (X^c_0, \dots, X^c_n) \leftarrow (O, \emptyset, \dots, \emptyset)$
                 // $X^c$ is clean w.r.t. $\beta^c$ and $X^c \cup X^0$ is clean w.r.t. $\beta^0$
3  $X^{-1} = (X^{-1}_0, \dots, X^{-1}_n) \leftarrow (O, \emptyset, \dots, \emptyset)$
   // Phase 2: Make $X^c$ a clean leximin allocation w.r.t. $\{\beta^c_h\}_{h \in N}$
4  repeat
5      while for some $i \in N$, there exists a Pareto-improving path from $F_{\beta^c_i}(X^c, i)$ in
        $\mathcal{G}^w(X^c, X^0, \beta^c)$ do
6          $P = (o'_1, \dots, o'_t) \leftarrow$ a min weight Pareto-improving path from $F_{\beta^c_i}(X^c, i)$ to
            $X^c_0$ in $\mathcal{G}^w(X^c, X^0, \beta^c)$
                                        /* Augment the allocation with the path $P$ */
7          $X^c_k \leftarrow X^c_k \Lambda P$ for all $k \in N + 0 - i$
8          $X^c_i \leftarrow X^c_i \Lambda P + o'_1$
9          $X^0_k \leftarrow X^0_k - o'_t$ for all $k \in N$
10         $X^0_0 \leftarrow X^0_0 + o'_t$
11     if for some $i \in N$, there exists an exchange path from $F_{\beta^c_i}(X^c, i)$ to some $X^c_j$
        such that either (a) $|X^c_i| < |X^c_j| + 1$ or (b) $|X^c_i| = |X^c_j| + 1$ and $i < j$ then
12         $P = (o'_1, \dots, o'_t) \leftarrow$ a min weight exchange path from $F_{\beta^c_i}(X^c, i)$ to $X^c_j$ in
            $\mathcal{G}^w(X^c, X^0, \beta^c)$       /* Augment the allocation with the path $P$ */
13         $X^c_k \leftarrow X^c_k \Lambda P$ for all $k \in N + 0 - i$
14         $X^c_i \leftarrow X^c_i \Lambda P + o'_1$
15 while at least one path augmentation was done in the iteration
   // Phase 3: Greedily allocate the items in $X^c_0 \cap X^0_0$
16 while $|X^c_0 \cap X^0_0 \cap X^{-1}_0| > 0$ do                /* Unallocated items exist */
17     $S \leftarrow \arg\max_{h \in N} v_h(X^c_h \cup X^0_h \cup X^{-1}_h)$     /* Set of all max-utility agents */
18     $i \leftarrow \max_{j \in S} j$                                /* Break ties using index */
19     $o \leftarrow$ an arbitrary item in $X^c_0 \cap X^0_0 \cap X^{-1}_0$
20     $X^{-1}_i \leftarrow X^{-1}_i + o$
21     $X^{-1}_0 \leftarrow X^{-1}_0 - o$
22 return $X^c \cup X^0 \cup X^{-1}$
```

augmenting along min weight Pareto-improving paths and min weight exchange paths until the allocation is leximin.

In the third phase, we update X^{-1} by allocating the remaining items (in $X^c_0 \cap X^0_0$). We do so greedily by allocating each item to the agent with the highest utility, under the assumption that these items have a marginal utility of -1. The exact steps are described in Algorithm 1.

We analyze each phase separately and establish key properties that the allocations X^c, X^0 and X^{-1} have at the end of each phase. In order to show computational efficiency, we use the value oracle model where we have oracle access to each agent's valuation function. A computationally efficient algorithm runs in polynomial time (in n and m) and only uses a polynomial number of queries to each value oracle.

5.1 Phase 1

The first phase is a simple setup where we allocate as many non-negative valued items as possible. X^0 is initialized as a clean `MAX-USW` allocation with respect to the valuations $\{\beta_h^0\}_{h\in N}$. X^c and X^{-1} are initialized as the empty allocation. Note that X^c is trivially clean with respect to the valuations $\{\beta_h^c\}_{h\in N}$.

Lemma 5.1. *At the end of Phase 1, X^c is clean with respect to $\{\beta_h^c\}_{h\in N}$ and $X^c \cup X^0$ is a `MAX-USW` clean allocation with respect to $\{\beta_h^0\}_{h\in N}$.*

The computational efficiency of Phase 1 relies on the computational efficiency of Yankee Swap. Yankee Swap uses a polynomial number of queries to $\{\beta_h^0\}_{h\in N}$. We can easily construct an efficient oracle for each β_h^0 to ensure Phase 1 runs in polynomial time and a polynomial number of valuation queries.

5.2 Phase 2

In this phase, we use path augmentations to manipulate X^c into a partial leximin allocation. There are three types of paths we check for and augment if it exists:

(a) A Pareto-improving path from $F_{\beta_i^c}(X^c, i)$ for some $i \in N$ to X_0^c in $\mathcal{G}^w(X^c, X^0, \beta^c)$.

(b) An exchange path from $F_{\beta_i^c}(X^c, i)$ for some $i \in N$ to some X_j^c such that $|X_i^c| < |X_j^c| + 1$.

(c) An exchange path from $F_{\beta_i^c}(X^c, i)$ for some $i \in N$ to some X_j^c such that $|X_i^c| = |X_j^c| + 1$ and $i < j$.

Note that these path augmentations work as intended since we maintain the invariant that $X^c \cup X^0$ is clean with respect to $\{\beta_h^0\}_{h\in N}$. Since exchange paths ((b) and (c)) are only guaranteed to work when there are no Pareto-improving paths, we ensure that we augment Pareto-improving paths (a) first before we check and augment exchange paths. This is clear in the steps of Algorithm 1.

Formally, let Z^c be a clean leximin allocation with respect to the valuations $\{\beta_h^c\}_{h\in N}$. If there are multiple, let Z^c be an allocation that is not lexicographically dominated (w.r.t. $\{\beta_h^c\}_{h\in N}$) by any other leximin allocation. Phase 2 ensures that for each agent $i \in N$, $|X_i^c| = |Z_i^c|$. We have the following Lemma.

Lemma 5.2. *Let X^c be a clean allocation with respect to $\{\beta_h^c\}_{h\in N}$ and X^0 be an allocation such that $X^c \cup X^0$ is clean with respect to $\{\beta_h^0\}_{h\in N}$. Let Z^c be a clean leximin allocation with respect to the valuations $\{\beta_h^c\}_{h\in N}$. If there are multiple, let Z^c be an allocation that is not lexicographically dominated (w.r.t. $\{\beta_h^c\}_{h\in N}$) by any other leximin allocation. Then there exists an agent $\ell \in N$ such that $|X_\ell^c| \neq |Z_\ell^c|$ if and only if at least one of the following conditions hold:*

(a) There exists a Pareto-improving path from $F_{\beta_i^c}(X^c, i)$ for some $i \in N$ to X_0^c in $\mathcal{G}^w(X^c, X^0, \beta^c)$.

(b) There exists an exchange path from $F_{\beta_i^c}(X^c, i)$ for some $i \in N$ to some X_j^c such that $|X_i^c| < |X_j^c| + 1$.
(c) There exists an exchange path from $F_{\beta_i^c}(X^c, i)$ for some $i \in N$ to some X_j^c such that $|X_i^c| = |X_j^c| + 1$ and $i < j$.

Lemma 5.2 immediately implies that *if* Phase 2 terminates, it terminates when $|X_h^c| = |Z_h^c|$ for all $h \in N$. Our next Lemma shows that Phase 2 indeed terminates in polynomial time. This result follows from bounding the number of path augmentations our algorithm computes.

Lemma 5.3. *Phase 2 terminates in polynomial time and polynomial valuation queries.*

Note that after each path transfer the number of allocated items in $X^c \cup X^0$ weakly increases. Therefore, $X^c \cup X^0$ remains a clean MAX-USW allocation at the end of Phase 2. Formally, we can make the following observation.

Lemma 5.4. *At the end of Phase 2, X^c is a clean lexicographically dominating leximin allocation with respect to the valuations $\{\beta_h^c\}_{h \in N}$ and $X^c \cup X^0$ is a MAX-USW clean allocation with respect to $\{\beta_h^0\}_{h \in N}$.*

5.3 Phase 3

At this point, $X^c \cup X^0$ is a MAX-USW clean allocation with respect to $\{\beta_h^0\}_{h \in N}$. Thus, all of the remaining items have a marginal value of -1 to every agent. It remains to assign these items in as equitable a manner as possible. In Phase 3, we sequentially allocate the remaining items, giving a "bad" item to the agent with the highest utility.

To carefully compare allocations, we adapt the comparison method of *domination* introduced in [12]. To compare two allocations $Y = Y^c \cup Y^0 \cup Y^{-1}$ and $\hat{Y} = \hat{Y}^c \cup \hat{Y}^0 \cup \hat{Y}^{-1}$, we first check the sorted utility vectors $\boldsymbol{s}^{Y^c}$ and $\boldsymbol{s}^{\hat{Y}^c}$; if $\boldsymbol{s}^{Y^c}$ lexicographically dominates $\boldsymbol{s}^{\hat{Y}^c}$, then we say Y dominates $\hat{Y}$. If $\boldsymbol{s}^{Y^c}$ and $\boldsymbol{s}^{\hat{Y}^c}$ are equal, we check the utility vectors $\boldsymbol{u}^{Y^c}$ and $\boldsymbol{u}^{\hat{Y}^c}$; if $\boldsymbol{u}^{Y^c}$ lexicographically dominates $\boldsymbol{u}^{\hat{Y}^c}$, then we say Y dominates $\hat{Y}$. If $\boldsymbol{u}^{Y^c}$ and $\boldsymbol{u}^{\hat{Y}^c}$ are equal as well, we check $\boldsymbol{u}^{Y}$ and $\boldsymbol{u}^{\hat{Y}}$; if $\boldsymbol{u}^{Y}$ lexicographically dominates $\boldsymbol{u}^{\hat{Y}}$, then we say Y dominates $\hat{Y}$.

Definition 5.5 (Domination). *An allocation $Y = Y^c \cup Y^0 \cup Y^{-1}$ dominates the allocation $\hat{Y} = \hat{Y}^c \cup \hat{Y}^0 \cup \hat{Y}^{-1}$ if any of the following conditions hold: (a) $\boldsymbol{s}^{Y^c} \succ_{\texttt{lex}} \boldsymbol{s}^{\hat{Y}^c}$, (b) $\boldsymbol{s}^{Y^c} = \boldsymbol{s}^{\hat{Y}^c}$ and $\boldsymbol{u}^{Y^c} \succ_{\texttt{lex}} \boldsymbol{u}^{\hat{Y}^c}$, or (c) $\boldsymbol{u}^{Y^c} = \boldsymbol{u}^{\hat{Y}^c}$ and $\boldsymbol{u}^{Y} \succ_{\texttt{lex}} \boldsymbol{u}^{\hat{Y}}$.*

Let $Y = Y^c \cup Y^0 \cup Y^{-1}$ be a *complete* leximin allocation for the original instance with valuations $\{v_h\}_{h \in N}$. If there are multiple leximin allocations, pick one which is not dominated by any other leximin allocation $\hat{Y}$. Due to this specific choice, we sometimes refer to Y as a dominating leximin allocation.

We first show that, just like X^c, Y^c is a lexicographically dominating leximin allocation with respect to the valuations $\{\beta_h^c\}_{h \in N}$.

Lemma 5.6. *After the end of Phase* 2, $|X_h^c| = |Y_h^c|$ *for all* $h \in N$.

Next, we show that X^c, X^0 and X^{-1} form a valid decomposition of $X^c \cup X^0 \cup X^{-1}$.

Lemma 5.7. *At every iteration in Phase* 3, *for any agent* $i \in N$, $v_i(X_i) = c|X_i^c| - |X_i^{-1}|$.

Combining these Lemmas, we can show the correctness of our algorithm.

Theorem 5.8. *When agents have* $\{-1, 0, c\}$*-ONSUB valuations, Algorithm 1 computes a leximin allocation efficiently.*

A useful corollary of this analysis is that Algorithm 1 outputs a `MAX-USW` allocation.

Corollary 5.9. *When agents have* $\{-1, 0, c\}$*-ONSUB valuations, Algorithm 1 outputs a* `MAX-USW` *allocation.*

6 Properties of the Leximin Allocation

Now that we have shown how a leximin allocation can be computed, we explore its connection to other fairness notions. Specifically, we study the following fairness notions.

Proportionality: An allocation X is said to be proportional if each agent receives at least an n-th fraction of their value for the entire set of items. This is not always possible — consider an instance with two agents and one high valued item. The fair allocation literature has therefore, instead, studied a relaxation of proportionality called proportionality up to one item [2]. An allocation X is *proportional up to one item* (`PROP1`) if any of the three following conditions hold for every agent $i \in N$: (a) $v_i(X_i) \geq \frac{1}{n} v_i(O)$, (b) $v_i(X_i + o) \geq \frac{1}{n} v_i(O)$ for some $o \in O \setminus X_i$, or (c) $v_i(X_i - o) \geq \frac{1}{n} v_i(O)$ for some $o \in X_i$.

Envy-freeness: An allocation is *envy-free* if no agent prefers another agent's bundle to their own. This, again, is impossible to guarantee when all items are allocated. We therefore, instead, look at the notion of *envy-freeness up to one item* (`EF1`) [10, 22, 2]. When there are both goods and chores, an allocation X is `EF1` if for any two agents i, j, there exists either some $o \in X_i$ such that $v_i(X_i - o) \geq v_i(X_j)$, or some $o \in X_j$ such that $v_i(X_i) \geq v_i(X_j - o)$.

Maximin Share: The *maximin share* (`MMS`) of an agent i is defined as the value they would obtain had they divided the items into n bundles themselves and picked the worst of these bundles. More formally,

$$\texttt{MMS}_i = \max_{X=(X_1, X_2, \dots, X_n)} \min_{j \in [n]} v_i(X_j).$$

[23] show that agents cannot always be guaranteed their maximin share; past works [21] instead focus on guaranteeing that every agent receives a fraction of their maximin share. For some $\varepsilon \in (0, 1]$, an allocation X is ε-`MMS` if for every agent $i \in N$, $v_i(X_i) \geq \varepsilon \cdot \texttt{MMS}_i$.

Lorenz Dominance: An allocation X is *Lorenz dominating* [4] if for all other allocations Y and all $k \in [n]$, it holds that $\sum_{j \in [k]} \boldsymbol{s}^X_j \geq \sum_{j \in [k]} \boldsymbol{s}^Y_j$ where $\boldsymbol{s}^X$ is the sorted utility vector of X (defined in Section 2.2).
Our main result is as follows,

Theorem 6.1. *When agents have $\{-1, 0, c\}$-ONSUB valuations, leximin allocations are guaranteed to be* `PROP1` *and Lorenz dominating. Additionally, when agents have $\{-1, 0, c\}$-ADD valuations, leximin allocations are guaranteed to be* `EF1` *and* 1-`MMS`.

7 NP-Hardness when *c* is not an Integer

In this section, we show that the problem of computing leximin allocations is NP-hard even for $\{-p, q\}$-ADD valuations for any co-prime integers p and q such that $p \geq 3$. This proof is very similar to the hardness result in [1]. Note that the assumption that p and q are co-prime is necessary since if p divides q, the problem reduces to computing a leximin allocation for agents with $\{-1, \frac{q}{p}\}$-ADD valuations and admits a polynomial time algorithm (Theorem 5.8). More generally, any common divisor of p and q can be eliminated by scaling agent valuations.

Theorem 7.1. *The problem of computing leximin allocations is NP-hard even when agents have $\{-p, q\}$-ADD valuations for any co-prime positive integers p and q such that $p \geq 3$.*

While Theorem 7.1 shows that the problem of computing leximin allocations is NP-hard for most values of p and q, there are two special cases which still remain unresloved — $\{-c, 0, 1\}$-ONSUB valuations and $\{-2, 0, c\}$-ONSUB valuations. We leave these two cases for future work.

8 Conclusions and Future Work

In this work, we study the computation of leximin allocations in instances with mixed goods and chores. We show that when agents have $\{-1, 0, c\}$-ONSUB valuations, leximin allocations can be computed efficiently. We also show that these allocations are Lorenz dominating and approximately proportional.

On a higher level, our work is the first to generalize the path augmentation technique to tri-valued valuation functions. We are hopeful that the tools of weighted exchange graphs and decompositions can be applied to even more general valuation classes. We are also excited by the class of order-neutral submodular valuations. Much like Rado and OXS valuations, we believe order-neutral submodular valuations are an appealing sub-class of submodular valuation functions that warrant further study.

Acknowledgements We thank anonymous WINE 2023 reviewers and Justin Payan for their useful comments. Viswanathan and Zick are funded by the National Science Foundation grant IIS-2327057.

References

1. Akrami, H., Chaudhury, B.R., Hoefer, M., Mehlhorn, K., Schmalhofer, M., Shahkarami, G., Varricchio, G., Vermande, Q., van Wijland, E.: Maximizing nash social welfare in 2-value instances. In: Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI) (2022)
2. Aziz, H., Caragiannis, I., Igarashi, A., Walsh, T.: Fair allocation of indivisible goods and chores. Autonomous Agents and Multi-Agent Systems **36**, 1–21 (2022)
3. Aziz, H., Lindsay, J., Ritossa, A., Suzuki, M.: Fair allocation of two types of chores. In: Proceedings of the 22nd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (2023)
4. Babaioff, M., Ezra, T., Feige, U.: Fair and truthful mechanisms for dichotomous valuations. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI). pp. 5119–5126 (2021)
5. Barman, S., Narayan, V., Verma, P.: Fair chore division under binary supermodular costs. In: Proceedings of the 22nd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (2023)
6. Barman, S., Verma, P.: Existence and computation of maximin fair allocations under matroid-rank valuations. In: Proceedings of the 20th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS). pp. 169–177 (2021)
7. Barman, S., Verma, P.: Truthful and fair mechanisms for matroid-rank valuations. In: Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI) (2022)
8. Benabbou, N., Chakraborty, M., Igarashi, A., Zick, Y.: Finding fair and efficient allocations for matroid rank valuations. ACM Transactions on Economics and Computation **9**(4) (2021)
9. Bhaskar, U., Sricharan, A.R., Vaish, R.: On approximate envy-freeness for indivisible chores and mixed resources. In: Proceedings of the Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021. pp. 1–23 (2021)
10. Budish, E.: The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. Journal of Political Economy **119**(6), 1061 – 1103 (2011)
11. Bérczi, K., Bérczi-Kovács, E.R., Boros, E., Gedefa, F.T., Kamiyama, N., Kavitha, T., Kobayashi, Y., Makino, K.: Envy-free relaxations for goods, chores, and mixed items (2020)
12. Cousins, C., Viswanathan, V., Zick, Y.: Dividing good and better items among agents with submodular valuations (2023). https://doi.org/10.48550/ARXIV.2302.03087, `https://arxiv.org/abs/2302.03087`
13. Ebadian, S., Peters, D., Shah, N.: How to fairly allocate easy and difficult chores. In: Proceedings of the 21st International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS). p. 372–380 (2022)
14. Garg, J., Murhekar, A.: Computing fair and efficient allocations with few utility values. In: Proceedings of the 14th International Symposium on Algorithmic Game Theory (SAGT). pp. 345–359. Springer International Publishing (2021)
15. Garg, J., Murhekar, A., Qin, J.: Fair and efficient allocations of chores under bivalued preferences. In: Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI) (2022)

16. Halpern, D., Procaccia, A.D., Psomas, A., Shah, N.: Fair division with binary valuations: One rule to rule them all. In: Proceedings of the 16th Conference on Web and Internet Economics (WINE). p. 370–383 (2020)
17. Hosseini, H., Sikdar, S., Vaish, R., Xia, L.: Fairly dividing mixtures of goods and chores under lexicographic preferences. In: Proceedings of the 22nd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (2023)
18. Kulkarni, R., Mehta, R., Taki, S.: Approximating maximin shares with mixed manna. In: Proceedings of the 21st ACM Conference on Economics and Computation (EC) (2021)
19. Kulkarni, R., Mehta, R., Taki, S.: On the ptas for maximin shares in an indivisible mixed manna. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI). pp. 5523–5530 (2021)
20. Kurokawa, D., Procaccia, A.D., Shah, N.: Leximin allocations in the real world. ACM Transactions of Economics and Computation **6**(3–4) (2018)
21. Kurokawa, D., Procaccia, A.D., Wang, J.: Fair enough: Guaranteeing approximate maximin shares. Journal of the ACM **65**(2) (2018)
22. Lipton, R.J., Markakis, E., Mossel, E., Saberi, A.: On approximately fair allocations of indivisible goods. In: Proceedings of the 5th ACM Conference on Economics and Computation (EC). p. 125–131 (2004)
23. Procaccia, A.D., Wang, J.: Fair enough: Guaranteeing approximate maximin shares. In: Proceedings of the 15th ACM Conference on Economics and Computation (EC). pp. 675–692 (2014)
24. Schrijver, A.: Combinatorial Optimization - Polyhedra and Efficiency. Springer (2003)
25. Viswanathan, V., Zick, Y.: A general framework for fair allocation under matroid rank valuations. In: Proceedings of the 24th ACM Conference on Economics and Computation (EC) (2023)
26. Viswanathan, V., Zick, Y.: Weighted notions of fairness with binary supermodular chores. arXiv preprint arXiv:2303.06212 (2023)
27. Viswanathan, V., Zick, Y.: Yankee swap: a fast and simple fair allocation mechanism for matroid rank valuations. In: Proceedings of the 22nd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (2023)