Dividing Good and Great Items among Agents with Bivalued Submodular Valuations

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 a set of indivisible goods among agents with *bivalued submodular valuations* — each good provides a marginal gain of either a or b ($a < b$) and goods have decreasing marginal gains. This is a natural generalization of two well-studied valuation classes — bivalued additive valuations and binary submodular valuations. We present a simple sequential algorithmic framework, based on the recently introduced Yankee Swap mechanism, that can be adapted to compute a variety of solution concepts, including max Nash welfare (MNW), leximin and p-mean welfare maximizing allocations when a divides b. This result is complemented by an existing result on the computational intractability of MNW and leximin allocations when a does not divide b. We show that MNW and leximin allocations guarantee each agent at least $\frac{2}{5}$ and $\frac{a}{b+2a}$ of their maximin share, respectively, when a divides b. We also show that neither the leximin nor the MNW allocation is guaranteed to be envy free up to one good (EF1). This is surprising since for the simpler classes of bivalued additive valuations and binary submodular valuations, MNW allocations are known to be envy free up to *any* good (EFX).

Keywords: Fair Allocation · Indivisible Goods · Submodular Valuations

1 Introduction

Fair allocation of indivisible goods has gained significant attention in recent years. The problem is simple: we need to assign a set of indivisible *goods* to a set of *agents*. Each agent has a subjective preference over the bundle of goods they receive. Our objective is to find an allocation that satisfies certain *fairness* and *efficiency* (or more generally *justice*) criteria. For example, some allocations maximize the product of agents' utilities, whereas others guarantee that no agent prefers another agent's bundle to their own. The fair allocation literature focuses on the existence and computation of allocations that satisfy a set of *justice criteria* (see [4] for a recent survey). For example, [26] focuses on computing *envy free up to any good* (EFX) allocations while [12] focuses on computing *max Nash welfare* allocations. Indeed, as these papers show, computing "fair" allocations without any constraint on agent valuations is an intractable problem for most justice criteria [26,12,23].

Justice Criterion	$\{0,1\}$**-SUB**	$\{1,c\}$**-ADD**	$\{1,c\}$**-SUB**	$\{a,b\}$**-ADD**
Nash Welfare	**P** [5]	**P**[1]	**P** (Theorem 2)	**NPh**[1]
Leximin	**P** [5]	**P** (Theorem 3)	**P** (Theorem 3)	**NPh** [1]
p-mean Welfare	**P** [5]	**P** (Theorem 4)	**P** (Theorem 4)	**NPh** [1]

Table 1: The computational complexity of computing various justice criteria under various valuation classes. a, b and c are positive integers. NPh is short for NP-hard. SUB is short for submodular and ADD is short for additive.

This has led to a recent systematic attempt to "push the computational envelope" — identify simpler classes of agent valuations for which there exist efficiently computable allocations satisfying multiple fairness and efficiency criteria. Several positive results are known when agents have *binary submodular* valuations; that is, their valuations exhibit diminishing returns to scale, and the added benefit of any good is either 0 or 1 [5,7,8,10,29,28]. Other works offer positive results when agents have *bivalued* additive preferences, i.e., each good has a value of either a or b, and agents' values for a bundle of goods is the sum of their utility for individual goods [1,3,18,19]. These results are encouraging in the face of known intractability barriers. Moreover, they offer practical benefits: binary submodular valuations naturally arise in settings such as course allocation [29], shift allocation [5] and in public housing assignment [9]. We take this line of work a step further and study a generalization of both bivalued additive valuations and binary submodular valuations.

1.1 Our Contributions

When agents have binary submodular valuations, *leximin* allocations are known to satisfy several desirable criteria. An allocation is leximin (or more precisely, leximin dominant) if it maximizes the welfare of the worst-off agent, then subject to that the welfare of the second worst-off agent and so on. When agents have binary submodular valuations, a leximin allocation maximizes utilitarian and Nash social welfare, is envy-free up to any good (EFX), and offers each agent at least $\frac{1}{2}$ of their maximin share [5]. Furthermore, there exists a simple sequential allocation mechanism that computes a leximin allocation [29].

We consider settings where agents have *bivalued submodular* valuations. That is, the marginal contribution of any good is either a or b, and marginal gains decrease as agents gain more goods. We assume that a divides b, or w.l.o.g. (by rescaling) that $a = 1$ and b is a positive integer. When agents have bivalued submodular valuations, unlike binary submodular valuations, leximin allocations no longer satisfy multiple fairness and efficiency guarantees (see Example 2). Thus, different justice criteria cannot be satisfied by a single allocation. To address this, we present a general, yet surprisingly simple algorithm (called Bivalued Yankee Swap) that efficiently computes allocations for a broad number of justice criteria. The algorithm (Algorithm 1) is based on the Yankee Swap protocol proposed by [29]: we start with all goods unassigned. At every round we select an agent (based on a selection criterion ϕ). If the selected agent can pick an unassigned good that offers them a high marginal benefit, they do so and we move on. Otherwise, we check whether they can *steal* a high-value good from another agent. We allow an agent to steal a high-value good if the agent who they want to steal from can recover their utility by either taking an unassigned good or stealing a good from another agent. This results in a *transfer path* where the initiating agent increases their utility by b, every other agent retains the same utility, and one good is removed from the pile of unassigned goods. If no such path exists, the agent is no longer allowed to "play" for high-value goods, and can either take low-value goods or none at all.

By simply modifying the agent selection criterion ϕ, this protocol outputs allocations that satisfy a number of "acceptable" justice criteria. We think of justice criteria as ways of comparing allocations; for example, an allocation X is better than an allocation Y according to the Nash-Welfare criterion if it either has fewer agents with zero utilities, or if the product of agent utilities under X is greater than the product of agent utilities under Y. A justice criterion Ψ is acceptable if (informally), it satisfies a notion of Pareto dominance, and if it admits a selection criterion (also referred to as a *gain function*) ϕ that decides which agent should receive a good in a manner consistent with Ψ (see Section 4.1 for details).

Several well-known justice criteria are acceptable (see Table 1). For every acceptable justice criterion, our result immediately implies a simple algorithmic framework that computes an allocation maximizing that justice criterion: one needs to simply implement Algorithm 1 with the appropriate gain function ϕ.

To complement our algorithmic results, we further analyze Nash welfare maximizing and leximin allocations. We show that neither are guaranteed to even be envy free up to one good (EF1). This result shows that bivalued submodular valuations signify a departure from both binary submodular and bivalued additive valuations, where the max Nash welfare allocation is guaranteed to be envy free up to any good (EFX). While envy-freeness is not guaranteed under bivalued submodular valuations, we do show that max Nash welfare and leximin allocations offer approximate MMS (maximin share) guarantees to agents. Specifically, we show that max Nash welfare allocations and leximin allocations are $\frac{2}{5}$-MMS and $\frac{a}{b+2a}$-MMS respectively.

1.2 Related Work

Our work is closely related to works on fair allocation under matroid rank (binary submodular) valuations. The problem of fair allocation under matroid rank valuations is reasonably well studied and has seen a surprising number of positive results. [10] shows that utilitarian welfare maximizing envy free up to one good (EF1) allocations always exist and can be computed efficiently. [7] shows that an MMS allocation is guaranteed to exist and can be computed efficiently. [5] shows that a Lorenz dominating allocation (which is both leximin and maximizes Nash welfare) is guaranteed to exist and can be computed efficiently. More recently, [28] presents a general framework (called General Yankee Swap) that can be used to compute weighted notions of fairness such as weighted max Nash welfare and weighted leximin efficiently, in addition to several others. Almost all of the papers in this field use path transfers in their algorithms [29,28,5,7]; a technique which we exploit as well. Our algorithm (called Bivalued Yankee Swap) is closely related to the General Yankee Swap framework [28]. However, due to the complexity of bivalued submodular valuations, the analysis in our setting is more involved.

Our paper also builds on results for fair allocation under bivalued additive valuations. [3] shows that the max Nash welfare allocation is envy free up to any good (EFX) when agents have bivalued additive valuations. [19] showed that an EFX and Pareto optimal allocation can be computed efficiently under bivalued preferences. [1] presents an algorithm to compute a max Nash welfare allocation efficiently when a divides b. This work is arguably the closest to ours. While their algorithm is different, the essential technical ingredients are the same; their algorithm uses transfer paths and decompositions as well. However, their analysis is restricted to bivalued additive valuations and the max Nash welfare allocation. Our results generalize their work both in terms of the class of valuations and the justice criteria considered. In a recent follow up work, [2] presents a polynomial time algorithm to compute a max Nash welfare allocation for the case where a divides $2b$. This case turns out to be surprisingly more complicated than the case where a divides b, requiring a different set of techniques to manipulate the high and low valued goods.

Bivalued additive valuations have also been studied in the realm of chores. [20] and [18] present efficient algorithms to compute an EF1 and Pareto optimal allocation when agents have bivalued preferences.

2 Preliminaries

We use $[t]$ to denote the set $\{1, 2, \ldots, t\}$. For ease of readability, we replace $A \cup \{g\}$ and $A \setminus \{g\}$ with $A + g$ and $A - g$ respectively.

We have a set of n *agents* $N = [n]$ and m *goods* $G = \{g_1, g_2, \ldots, g_m\}$. Each agent $i \in N$ has a *valuation function* $v_i : 2^G \mapsto \mathbb{R}_{\geq 0}$; $v_i(S)$ specifies agent i's value for the bundle S. Given a valuation function v, we let $\Delta_v(S, g) = v(S + g) - v(S)$ denote the marginal utility of the good g to the bundle S under v. When clear from context, we write $\Delta_i(S, g)$ instead of $\Delta_{v_i}(S, g)$ to denote the marginal utility of giving the good g to agent i given that they have already been assigned the bundle S.

Given $a, b \in \mathbb{R}_{\geq 0}$ such that $a < b$, we say that v_i is an (a, b)-*bivalued submodular valuation* if v_i satisfies the following three properties: (a) $v_i(\emptyset) = 0$, (b) $\Delta_i(S, g) \in \{a, b\}$ for all $S \subset G$ and $g \in G \setminus S$, and (c) $\Delta_i(S, g) \geq \Delta_i(T, g)$ for all $S \subseteq T \subset G$ and $g \in G \setminus T$. We use $\{a, b\}$-SUB to denote (a, b)-bivalued submodular valuation functions.

A lot of our analysis uses the class of $\{0, 1\}$-SUB valuations. $\{0, 1\}$-SUB valuations are also known as *binary submodular valuations*, and have been extensively studied (see Section 1.2 for a discussion). When $a = 0$, the valuation function

v_i is essentially a scaled binary submodular valuation; specifically, $\frac{1}{b}v_i(\cdot)$ is a $\{0,1\}$-SUB valuation. Existing results under $\{0,1\}$-SUB valuations trivially extend to $\{0,b\}$-SUB valuations as well, and generally offer stronger guarantees than the ones offered in this work. Thus, we focus our attention on the case where $b > a > 0$.

For ease of readability, we scale valuations by a. Under this scaling, all agents have $\{1, \frac{b}{a}\}$-SUB valuations. We further simplify notation and replace the value $\frac{b}{a}$ by c; under this notation, all agents have $\{1,c\}$-SUB valuations where $c > 1$. The value of c is the same for all agents. Note that when a divides b, c is a *natural number*.

An *allocation* $X = (X_0, X_1, X_2, \dots, X_n)$ is a partition of G into $n+1$ bundles. Each X_i denotes the bundle allocated to agent i; X_0 denotes the unallocated goods. Our goal is to compute *complete* allocations — allocations where $X_0 = \emptyset$. We refer to the value $v_i(X_i)$ as the *utility* of agent i under the allocation X and we define the *utility vector* of an allocation X (denoted by $\boldsymbol{u}^X$) as the vector $(v_1(X_1), v_2(X_2), \dots, v_n(X_n))$.

For ease of analysis, we sometimes refer to 0 as a dummy agent with valuation function $v_0(S) = c|S|$ and allocated bundle X_0. This is trivially a $\{1,c\}$-SUB function. None of our justice criteria consider agent 0. Our analysis will also use the following definition of lexicographic dominance.

Definition 1 (Lexicographic Dominance). *A vector $\boldsymbol{y} \in \mathbb{R}^n_{\geq 0}$ is said to* lexicographically dominate *another vector $\boldsymbol{z} \in \mathbb{R}^n_{\geq 0}$ if there exists some $k \in [n]$ such that for all $j \in [k-1]$, $y_j = z_j$ and $y_k > z_k$. This is denoted by $\boldsymbol{y} \succ_{\texttt{lex}} \boldsymbol{z}$. An allocation X is said to* lexicographically dominate *another allocation Y if $\boldsymbol{u}^X \succ_{\texttt{lex}} \boldsymbol{u}^Y$.*

2.1 Justice Criteria

We consider three central justice criteria.

Leximin: An allocation X is *leximin* if it maximizes the least utility in the allocation and subject to that, maximizes the second lowest utility and so on. This can be 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 $(v_1(X_1), v_2(X_2), \dots, v_n(X_n))$ *sorted in ascending order*. An allocation X is leximin if, for no other allocation Y, we have $\boldsymbol{s}^Y \succ_{\texttt{lex}} \boldsymbol{s}^X$.

Max Nash Welfare (MNW): Let the set of agents who receive a positive utility under an allocation X be denoted by P_X. An allocation X maximizes Nash welfare if it first maximizes the number of agents who receive a positive utility $|P_X|$ and subject to that, maximizes the value $\prod_{i \in P_X} v_i(X_i)$ [12].

p-Mean Welfare: The p-mean welfare of an allocation X is defined as $(\frac{1}{n}\sum_{i \in N} v_i(X_i)^p)^{1/p}$ for $p \leq 1$. Since this value is undefined when $v_i(X_i) = 0$ for any $i \in N$ and $p < 0$, we modify the definition slightly when defining a max p-mean welfare allocation. We again denote P_X as the set of agents who receive a positive utility under the allocation X. An allocation X is said to be a max

p-mean welfare allocation for any $p \leq 1$ if the allocation maximizes the size of P_X and subject to that, maximizes $\left(\frac{1}{n}\sum_{i \in P_X} v_i(X_i)^p\right)^{1/p}$.

The p-mean welfare functions have been extensively studied in economics [25], fair machine learning [22,15,14,16,17], and, more recently, fair allocation [6]. When p approaches $-\infty$, the p-mean welfare corresponds to the leximin objective and when p approaches 0, the p-mean welfare corresponds to the max Nash welfare objective.

Leximin dominance, Nash welfare and p-mean welfare are three ways of comparing allocations. More generally, we can think of a *justice criterion* Ψ as a way of comparing two allocations X and Y. Notice that we are specifically comparing the utility vectors of allocations. Therefore, we say allocation X is better than Y according to Ψ if $\boldsymbol{u}^X \succ_\Psi \boldsymbol{u}^Y$. To ensure that there indeed exists a Ψ maximizing allocation, we require that $\succeq_\Psi$ be a total ordering over all allocations. An allocation X is maximal with respect to Ψ if it is not Ψ-dominated by any other allocation, i.e., for any other allocation Y, it is not the case that $\boldsymbol{u}^Y \succ_\Psi \boldsymbol{u}^X$. For ease of readability, we sometimes abuse notation and use $X \succ_\Psi Y$ to denote $\boldsymbol{u}^X \succ_\Psi \boldsymbol{u}^Y$.

3 Bivalued Submodular Valuations

We now present some useful properties of bivalued submodular valuations that we use in our algorithms. Informally, we show that the number of high valued goods in any agent's bundle corresponds to a binary submodular function. This allows us to use existing techniques from the fair allocation under binary submodular functions literature in our analysis.

More formally, for each agent $i \in N + 0$, we define a valuation function $\beta_i : 2^G \mapsto \mathbb{R}_{\geq 0}$ as follows: for any $S \subseteq G$, $\beta_i(S)$ is equal to the size of the largest subset T of S such that $v_i(T) = c|T|$; that is, $\beta_i(S) = \max\{|T| : T \subseteq S, v_i(T) = c|T|\}$. In other words, all the goods in T provide a marginal gain of c to i. $\beta_i(S)$ captures the number of goods in S that provide a value of c to i. We show that $\beta_i \in \{0, 1\}$-SUB.

Lemma 1. *For each agent $i \in N + 0$ when v_i is a $\{1, c\}$-SUB valuation, β_i is a binary submodular valuation.*

Using β_i, we can leverage the properties of binary submodular valuations used in the fair allocation literature. However, these properties require bundles to be *clean* as well, as given by the following definition.

Definition 2 (Clean). *For any agent $i \in N + 0$, a bundle S is* clean *with respect to the binary submodular valuation β_i if $\beta_i(S) = |S|$. By definition, this is equivalent to saying $v_i(S) = c|S|$. An allocation X is said to be clean if for all agents $i \in N + 0$, $\beta_i(X_i) = |X_i|$. By our definition of v_0, any bundle S is clean with respect to the dummy agent 0.*

This is similar to the notion of non-wastefulness used in [1] and cleanness used in [10].

When agents have binary submodular valuations, given a clean allocation X, we define the *exchange graph* $\mathcal{G}(X)$ as a directed graph over the set of goods. An edge exists from g to g' in the exchange graph if $g \in X_j$ and $\beta_j(X_j - g + g') = \beta_j(X_j)$ for some $j \in N+0$. Since our problem instances have *bivalued* submodular valuations, whenever we refer to the exchange graph of an allocation, we refer to it with respect to the *binary* submodular valuations $\{\beta_i\}_{i \in N}$.

Let $P = (g_1, g_2, \dots, g_t)$ be a path in the exchange graph for a clean allocation X. We define a transfer of goods along the path P in the allocation X as the operation where g_t is given to the agent who has g_{t-1}, g_{t-1} is given to the agent who has g_{t-2} and so on till finally g_1 is discarded. We call this transfer *path augmentation*; the bundle X_i after path augmentation with the path P is denoted by $X_i \Lambda P$ and defined as $X_i \Lambda P = (X_i - g_t) \oplus \{g_j, g_{j+1} : g_j \in X_i\}$, where $\oplus$ denotes the symmetric set difference operation.

For any clean allocation X and agent i, we define $F_i(X) = \{g \in G : \Delta_{\beta_i}(X_i, g) = 1\}$ as the set of goods which give agent i a marginal gain of 1 under the valuation β_i, i.e., the set of all goods that give agent i a marginal gain of c under v_i. For any agent i, let $P = (g_1, \dots, g_t)$ be the shortest path from $F_i(X)$ to X_j for some $j \neq i$. Then path augmentation with the path P and giving g_1 to i results in an allocation where i's value for their bundle goes up by c, j's value for their bundle goes down by c and all the other agents do not see any change in value. This is formalized below and exists in [7, Lemma 1] and [29, Lemma 3.7].

Lemma 2 ([7], [29]). *Let X be a clean allocation with respect to the binary submodular valuations $\{\beta_i\}_{i \in N+0}$. Let $P = (g_1, \dots, g_t)$ be the shortest path in the exchange graph $\mathcal{G}(X)$ from $F_i(X)$ to X_j for some $i \in N+0$ and $j \in N+0-i$. Define the allocation Y as follows*

$$Y_k = \begin{cases} X_k \Lambda P & k \neq i \\ X_i \Lambda P + g_1 & k = i. \end{cases}$$

Then, we have for all $k \in N - i - j$, $\beta_k(Y_k) = \beta_k(X_k)$, $\beta_i(Y_i) = \beta_i(X_i) + 1$ and $\beta_j(Y_j) = \beta_j(X_j) - 1$. Furthermore, the new allocation Y is clean.

We now establish sufficient conditions for a path to exist. We say there is a path from some agent i to some agent j in an allocation X if there is a path from $F_i(X)$ to X_j in the exchange graph $\mathcal{G}(X)$. The following lemma appears in [29, Theorem 3.8].

Lemma 3 ([29]). *Let X and Y be two clean allocations with respect to the binary submodular valuations $\{\beta_i\}_{i \in N+0}$. For any agent i such that $|X_i| < |Y_i|$, there is a path in the exchange graph $\mathcal{G}(X)$ from $F_i(X)$ to some good in X_k for some $k \in N+0$ such that $|X_k| > |Y_k|$.*

Unfortunately, unlike binary submodular valuations [10], we cannot make any allocation clean without causing a loss in utility to some agents. We can however, efficiently *decompose* any allocation into a clean allocation and a *supplementary* allocation. For any allocation X, the clean part is denoted using X^c (since each good provides a value of c in a clean allocation) and the supplementary allocation is denoted using X^1. The supplementary allocation is a tuple of n disjoint bundles of G, that is $X^1 = (X_1^1, X_2^1, \dots, X_n^1)$.

The supplementary allocation is not technically an allocation since it is not an $n+1$ partition of G; we call it an allocation nevertheless to improve readability.

Example 1. Consider an example with two agents $\{1,2\}$ and four goods $\{g_1, \dots, g_4\}$. Agent valuations are given as follows

$$v_1(S) = c|S| \qquad\qquad v_2(S) = c\min\{|S|, 1\} + \max\{|S| - 1, 0\}$$

Consider an allocation X where $X_1 = \{g_1, g_2\}$ and $X_2 = \{g_3, g_4\}$. Note that both goods in X_1 give agent 1 a value of c but only one good in X_2 gives agent 2 a value of c. A natural decomposition in this case would be

$$\begin{aligned} X_0^c &= \{g_4\} & X_1^c &= \{g_1, g_2\} & X_2^c &= \{g_3\} \\ & & X_1^1 &= \emptyset & X_2^1 &= \{g_4\} \end{aligned}$$

X^c is the clean part and X^1 is the supplementary part. Note that, decompositions need not be unique. Swapping g_4 and g_3 in the above allocation would result in another decomposition.

Formally, X^c and X^1 is a decomposition of X if for all agents $i \in N$, we have (a) $X_i^1, X_i^c \subseteq X_i$, (b) $X_i^c \cap X_i^1 = \emptyset$, (c) $X_i^c \cup X_i^1 = X_i$, and (d) $|X_i^c| = \beta_i(X_i)$.

We show that a decomposition always exists.

Lemma 4. *For any allocation X, there exists a decomposition into a clean allocation X^c and a supplementary allocation X^1 such that for all $i \in N$, we have (a) $X_i^1, X_i^c \subseteq X_i$, (b) $X_i^c \cap X_i^1 = \emptyset$, (c) $X_i^c \cup X_i^1 = X_i$, and (d) $|X_i^c| = \beta_i(X_i)$.*

Note that by design, X_0^c contains all the goods in X^1. This is done to ensure X^c is an $n+1$ partition of G. We do not need to do this for X^1.

We define the union of a clean allocation X^c and a supplementary allocation X^1 (denoted $X^c \cup X^1$) as follows: for each agent $i \in N$, $X_i = X_i^c \cup X_i^1$ and $X_0 = G \setminus \bigcup_{i \in N} X_i$. This definition holds for any pair of clean and supplementary allocations and not just decompositions via Lemma 4. It is easy to see that if an allocation X was decomposed into X^c and X^1 (satisfying the properties of Lemma 4), then $X = X^c \cup X^1$. We will refer to an allocation X as $X = X^c \cup X^1$ to denote a decomposition of the allocation (via Lemma 4) into a clean allocation X^c and a supplementary allocation X^1. As we saw in Example 1, decompositions need not be unique. When we use $X = X^c \cup X^1$, we will refer to any one of the possible decompositions. The exact one will not matter.

Finally, we present a useful metric to compare allocations using their decompositions. We refer to this metric as *domination.* To compare two allocations X and Y, we first compare the sorted utility vectors of X^c and Y^c. If the sorted utility vector X^c lexicographically dominates that of Y^c, then X dominates Y;

if the two allocations X^c and Y^c have the same sorted utility vectors, we compare their utility vectors. If X^c is lexicographically greater than Y^c, then X dominates Y. If X^c and Y^c have the same utility vectors, we compare X and Y lexicographically. This definition is formalized below. Recall that $\boldsymbol{u}^X$ and $\boldsymbol{s}^X$ denote the utility vector and sorted utility vector of the allocation X respectively.

Definition 3 (Domination). *We say an allocation $X = X^c \cup X^1$* dominates *an allocation $Y = Y^c \cup Y^1$ if any of the following three conditions hold:*

(a) $\boldsymbol{s}^{X^c} \succ_{\texttt{lex}} \boldsymbol{s}^{Y^c}$
(b) $\boldsymbol{s}^{X^c} = \boldsymbol{s}^{Y^c}$ *and* $\boldsymbol{u}^{X^c} \succ_{\texttt{lex}} \boldsymbol{u}^{Y^c}$
(c) $\boldsymbol{u}^{X^c} = \boldsymbol{u}^{Y^c}$ *and* $\boldsymbol{u}^{X} \succ_{\texttt{lex}} \boldsymbol{u}^{Y}$

An allocation X is a dominating *Ψ maximizing allocation if no other Ψ maximizing allocation Y dominates X.*

4 Bivalued Yankee Swap

We now present *Bivalued Yankee Swap* — a flexible framework for fair allocation under $\{1, c\}$-SUB valuations. The results in this section assume that c is a natural number; in other words, we are interested in $\{a, b\}$-SUB valuations where a divides b.

In the original Yankee Swap algorithm [29], all goods start off initially unallocated. The algorithm proceeds in rounds; at each round, we select an agent based on some criteria. This agent can either take an unallocated good, or initiate a transfer path by stealing a good from another agent, who then steals a good from another agent and so on until some agent steals a good from the pool of unallocated goods. [29] shows that these transfer paths can be efficiently computed and are equivalent to paths on the exchange graph. If there is no transfer path from the agent to an unassigned good, the agent is removed from the game. We continue until all goods have been assigned. This algorithm can also be thought of as a variant of the classical matroid path augmentation algorithm where paths are chosen more carefully so as to maximize a specific justice criterion.

The Bivalued Yankee Swap algorithm is a modified version of this approach. We start by letting agents run Yankee Swap, but require that whenever an agent receives a good (by either taking an unassigned good or by stealing a good from another agent), that good must offer them a marginal gain of c. In addition, we require that every agent who had a good stolen from them fully recovers their utility, i.e., an agent who lost a good of marginal value c, must receive a good of marginal value c in exchange; an agent who lost a good of marginal value 1 must receive a good of marginal value 1 in exchange. Thus, whenever an agent initiates a transfer path, that path results in them receiving an additional utility of c, while all other agents' utilities remain the same. More formally, every agent starts in the set U. If the agent is able to find a path in the exchange graph to an unallocated good, we augment the allocation using this path. If the agent is unable find a path, we remove them from U. Once the agent is removed from U, no transfer path exists that can give the agent a value of c. Therefore, if an agent outside of U is chosen, we *provisionally* give the agent an arbitrary unassigned good, offering them a marginal gain of 1. Since all unassigned goods offer a marginal gain of 1 for every agent not in U, the choice of which good to allocate to guarantee a value of 1 does not matter. Therefore, we treat this provisionally allocated good as an unallocated good as well; thereby allowing transfer paths to steal this good. If this provisionally allocated good gets stolen, we replace it with another low-value good. The algorithm stops when there are no unallocated goods left. These steps are described in Algorithm 1.

We use a *gain function* ϕ to pick the next agent to invoke a transfer path. Informally, ϕ computes the change in the justice criterion Ψ when a good is assigned to an agent. The gain function concept is used to compute allocations satisfying a diverse set of justice criteria in [28], when agents have binary submodular valuations.

More formally, ϕ maps a tuple $(\boldsymbol{u}^X, i, d)$ consisting of the utility vector of an allocation X, an agent i and a value $d \in \{1, c\}$ to a finite real value. The value $\phi(\boldsymbol{u}^X, i, d)$ quantifies the gain in the justice criterion Ψ of adding a value of d to the agent i given the allocation X. We abuse notation and use $\phi(X, i, d)$ to denote $\phi(\boldsymbol{u}^X, i, d)$.

At each round, we find the agent $i \in U$ who maximizes $\phi(X, i, c)$ (implicitly assuming that i can find a good with a marginal gain of c), and the agent $j \in N \setminus U$ who maximizes $\phi(X, j, 1)$ (implicitly assuming that all goods offer j a marginal gain of 1). We then choose the agent, among i and j, who maximizes ϕ; in the algorithm, these values are denoted by $\texttt{Gain}_c$ and $\texttt{Gain}_1$. Ties are broken first in favor of agents in U and second, in favor of agents with a lower index. Let us present an example of how Algorithm 1 works, with two candidate gain functions: Example 2 shows how a leximin allocation and an `MNW` allocation are computed for a simple two-agent instance.

Example 2. Consider a setting with six goods $G = \{g_1, \dots, g_6\}$, and two additive agents 1 and 2. Agent 1's valuation function is $v_1(S) = |S|$, and Agent 2's valuation is $v_2(S) = 5|S|$. In other words, Agent 1 values every good at 1, whereas Agent 2 values every good at 5. We set the gain function to be $\phi_{\texttt{leximin}}(X, i, d) = -6v_i(X_i) + d$, to compute a leximin allocation (as per Theorem 3), and run Algorithm 1. All agents are initially in U, and we compute $\texttt{Gain}_c, \texttt{Gain}_1$. In the first iteration, $\texttt{Gain}_1 = -\infty$ since all agents are in U and $\texttt{Gain}_c = -6 \times v_1(\emptyset) + 5 = 5$. Both agents 1 and 2 have equal ϕ values so we break ties and choose Agent 1. However, there is no way to give Agent 1 a good which gives them a marginal gain of 5. We therefore remove Agent 1 from U.

In the next iteration, $\texttt{Gain}_c = -6 \times v_2(\emptyset) + 5 = -6 \times 0 + 5 = 5$, and $\texttt{Gain}_1 = -6 \times v_1(\emptyset) + 1 = 1$. Since $\texttt{Gain}_c > \texttt{Gain}_1$, we choose Agent 2 to receive a good. Since they value all the goods at 5, we pick an arbitrary good (say g_1) and allocate it to them.

In the next iteration, $\texttt{Gain}_c = -6 \times v_2(g_1) + 5 = -6 \times 5 + 5 = -25$ but $\texttt{Gain}_1 = -6 \times v_1(\emptyset) + 1 = 1$ still. So we have $\texttt{Gain}_1 > \texttt{Gain}_c$ and we choose agent 1 to

ALGORITHM 1: Bivalued Yankee Swap

```
Input  : A set of goods $G$, a set of agents $N$ with $\{1, c\}$-SUB valuations $\{v_h\}_{h \in N}$
         and a gain function $\phi$.
Output: A dominating $\Psi$ maximizing allocation.
1  $X^c = (X^c_0, X^c_1, \ldots, X^c_n) \leftarrow (G, \emptyset, \ldots, \emptyset)$      /* $X^c$ has no goods allocated. */
   // Invariant: $X^c_0$ stores the provisionally allocated goods as well as
      the unallocated goods.
2  $X^1 = (X^1_1, \ldots, X^1_n) \leftarrow (\emptyset, \ldots, \emptyset)$   /* Stores the provisional allocation. */
3  $U \leftarrow N$           /* Set of agents still in play for $c$ valued goods. */
4  while $\sum_{h \in N} |X^1_h| < |X^c_0|$ do            /* While unallocated goods exist. */
5      $\texttt{Gain}_c = \begin{cases} \max_{k \in U} \phi(X^c \cup X^1, k, c) & U \neq \emptyset \\ -\infty & U = \emptyset \end{cases}$
6      $\texttt{Gain}_1 = \begin{cases} \max_{k \in N \setminus U} \phi(X^c \cup X^1, k, 1) & N \setminus U \neq \emptyset \\ -\infty & N \setminus U = \emptyset \end{cases}$
7      if $\texttt{Gain}_c \geq \texttt{Gain}_1$ then
           // Try to give an agent from $U$ a value of $c$.
8          $S \leftarrow \arg\max_{k \in U} \phi(X^c \cup X^1, k, c)$
9          $i \leftarrow \min_{j \in S} j$
10         if a path in the exchange graph $\mathcal{G}(X^c)$ from $F_i(X^c)$ to $X^c_0$ exists then
11             $P = (g'_1, g'_2, \ldots, g'_k) \leftarrow$ the shortest path from $F_i(X^c)$ to $X^0$ in $\mathcal{G}(X^c)$
               // Augment the allocation with the path $P$ and give $g'_1$ to $i$
12             $X^c_k \leftarrow X^c_k \Lambda P$ for all $k \in N + 0 - i$
13             $X^c_i \leftarrow X^c_i \Lambda P + g'_1$
14             if $g'_k \in X^1_j$ for some $j \in N$ then
                   // Replace good stolen from $X^1_j$ with an arbitrary
                      unallocated good
15                 $X^1_j \leftarrow X^1_j - g'_k + g$ for some $g \in X^c_0 \setminus \bigcup_{h \in N} X^1_h$
16         else
17             $U \leftarrow U - i$      /* $i$ is no longer in play for $c$ valued goods. */
18     else if $\texttt{Gain}_c < \texttt{Gain}_1$ then
           // Give an agent from $N \setminus U$ a value of $1$.
19         $S \leftarrow \arg\max_{k \in N \setminus U} \phi(X^c \cup X^1, k, 1)$
20         $i \leftarrow \min_{j \in S} j$
21         $X^1_i \leftarrow X^1_i + g$ for some $g \in X^c_0 \setminus \bigcup_{h \in N} X^1_h$
22 return $X^c \cup X^1$
```

provisionally receive an arbitrary good (say g_2). We will have $\texttt{Gain}_1 > \texttt{Gain}_c$ for the remaining iterations as well, yielding the allocation $X_1 = \{g_2, \ldots, g_6\}, X_2 = \{g_1\}$, which is indeed leximin.

By modifying the gain function ϕ to be

$$\phi_{\texttt{MNW}}(X, i, d) = \begin{cases} \frac{v_i(X_i)+d}{v_i(X_i)} & v_i(X_i) > 0 \\ Md & v_i(X_i) = 0 \end{cases},$$

as per Theorem 2 (where M is a very large number), Algorithm 1 outputs an MNW allocation. When no goods are assigned, the first iteration proceeds the exact same way as that of the leximin allocation and Agent 1 is removed from U.

In the second iteration, we have $\texttt{Gain}_c = 5 \times M$ and $\texttt{Gain}_1 = M$. Thus, Agent 2 receives a good (say g_1). Next, we have that $\texttt{Gain}_c = \phi(X, 2, c) = \frac{v_2(g_1)+c}{v_2(g_1)} = \frac{2c}{c} = 2$, and $\texttt{Gain}_1 = \phi(X, 1, 1) = M \times 1 = M$. Now we have $\texttt{Gain}_1 > \texttt{Gain}_c$, so Agent 1 gets g_2. In the next iteration, we still have $\texttt{Gain}_c = \phi(X, 2, c) = \frac{2c}{c} = 2$, but $\texttt{Gain}_1 = \phi(X, 1, 1) = \frac{v_1(g_2)+1}{v_1(g_2)} = \frac{2}{1} = 2$ as well. Thus, according to our tiebreaking scheme, we let agent 2 pick a good next, and they get g_3. We continue in a similar manner, and end up with the allocation $X_1 = \{g_2, g_4, g_6\}, X_2 = \{g_1, g_3, g_5\}$, which is indeed MNW.

This example also shows how, unlike the binary submodular valuations case, max Nash welfare and leximin allocations can have significantly different sorted utility vectors.

4.1 When Does Bivalued Yankee Swap Work?

Bivalued Yankee Swap computes a Ψ-maximizing allocation when the following conditions are satisfied. The conditions are defined for any general vector $\boldsymbol{x}$ but it would help to think of $\boldsymbol{x}, \boldsymbol{y}$ and $\boldsymbol{z}$ as utility vectors.

(C1) – **Symmetric Pareto Dominance** Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^n_{\geq 0}$ be two vectors. Let $s(\boldsymbol{x})$ denote the vector $\boldsymbol{x}$ sorted in increasing order. If for all $i \in N$, $s(\boldsymbol{x})_i \geq s(\boldsymbol{y})_i$, then $\boldsymbol{x} \succeq_\Psi \boldsymbol{y}$. Equality holds if and only if $s(\boldsymbol{x}) = s(\boldsymbol{y})$.

(C2) – **Gain Function** Ψ admits a finite valued gain function ϕ that satisfies the following properties:

- **(G1)** Let $\boldsymbol{x} \in \mathbb{Z}^n_{\geq 0}$ be some vector, let $i, j \in N$ be two agents, and d_1, d_2 be two values in $\{1, c\}$. Let $\boldsymbol{y} \in \mathbb{Z}^n_{\geq 0}$ be the vector resulting from adding a value of d_1 to x_i. Let $\boldsymbol{z}$ be the vector resulting from adding a value of d_2 to x_j. If $\phi(\boldsymbol{x}, i, d_1) \geq \phi(\boldsymbol{x}, j, d_2)$, we must have $\boldsymbol{y} \succeq_\Psi \boldsymbol{z}$. Equality holds if and only if $\phi(\boldsymbol{x}, i, d_1) = \phi(\boldsymbol{x}, j, d_2)$.
- **(G2)** For any two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^n_{\geq 0}$, an agent $i \in N$ such that $x_i \leq y_i$ and any $d \in \{1, c\}$, we must have $\phi(\boldsymbol{x}, i, d) \geq \phi(\boldsymbol{y}, i, d)$. Equality holds if $x_i = y_i$.
- **(G3)** For any vector $\boldsymbol{x} \in \mathbb{Z}^n_{\geq 0}$ and any two agents $i, j \in N$, if $x_i \leq x_j$, then $\phi(\boldsymbol{x}, i, d) \geq \phi(\boldsymbol{x}, j, d)$ for any $d \in \{1, c\}$. Equality holds if and only if $x_i = x_j$.

There are two differences between our conditions and the conditions of the General Yankee Swap algorithm [28]. First, we strengthen Pareto Dominance to Symmetric Pareto Dominance (C1). Symmetric Pareto Dominance is not biased towards any agent and therefore, two allocations with the same sorted utility vector have the same objective value. As an immediate corollary, weighted notions of fairness like the max weighted Nash welfare objective [13] do not satisfy Symmetric Pareto Dominance (C1); that is, when agents have weights, two allocations with the same sorted utility vector may not have the same objective value. Second, we introduce (G3) which further strengthens our conditions; (G3) states that all things being equal, it is better to increase the utility of lower utility agents than higher utility agents. We conjecture that the justice criteria satisfying the conditions (C1) and (C2) correspond exactly to the set of generalized welfare functions [25, Chapter 3] but are unable to provide a proof. We leave this question to future work.

Our main result is the following.

Theorem 1. *Suppose that all agents in N have $\{1, c\}$-SUB valuations. Let Ψ be a justice criterion that satisfies (C1) and (C2), with a gain function ϕ. Bivalued Yankee Swap with the gain function ϕ outputs a dominating Ψ-maximizing allocation.*

To prove this, we first show an important Lemma stating that for any agent $i \in N$, if there is a path in the exchange graph $\mathcal{G}(Y^c)$ from $F_i(Y^c)$ to Y^c_j such that $|Y^c_i| < |Y^c_j| - 1$, then Y is not a dominating Ψ maximizing allocation. We then use this Lemma to show that when an agent i is removed from U, it happens at the right time. The main idea is somewhat similar to that of General Yankee Swap [28] but the arguments require a significantly more careful analysis.

5 Applying Bivalued Yankee Swap

We now turn to applying Theorem 1 to well known fairness objectives. While we do not prove this explicitly, in all cases, the gain function ϕ can be trivially computed in time $O(\tau)$ (where τ is an upper bound on the time to compute $v_i(S)$ for any i and any S).

5.1 Max Nash Welfare

Recall that a max Nash Welfare allocation X is one that maximizes the number of agents $|P_X|$ who receive a non-zero utility, and subject to that maximizes the product $\prod_{i \in P_X} v_i(X_i)$.

Theorem 2. *When Ψ corresponds to the Nash welfare, Bivalued Yankee Swap run with the following gain function $\phi_{\texttt{MNW}}$ computes a Nash welfare maximizing allocation.*

$$\phi_{\texttt{MNW}}(X, i, d) = \begin{cases} \frac{v_i(X_i)+d}{v_i(X_i)} & v_i(X_i) > 0 \\ Md & v_i(X_i) = 0 \end{cases}$$

where M is a very large positive number.

5.2 Leximin

Recall that a leximin allocation is one that maximizes the utility of the worst off agent, subject to that, maximizes the utility of the second worst off agent, and so on.

Theorem 3. *When Ψ corresponds to the leximin fairness objective, Bivalued Yankee Swap run with the gain function $\phi_{\texttt{leximin}}(X, i, d) = -(c+1)v_i(X_i) + d$ computes a leximin allocation.*

5.3 p-mean Welfare

Recall that the max p-mean welfare allocation X first maximizes the number of agents who receive a positive utility $|P_X|$ and subject to that, maximizes $M_p(X) = \left(\frac{1}{n}\sum_{i \in P_X} v_i(X_i)^p\right)^{1/p}$ where $p \leq 1$. We have already shown how to compute this justice criterion for certain p values: M_0 corresponds to Nash welfare (Section 5.1) and $M_{-\infty}$ corresponds to leximin (Section 5.2). We now show how to compute a p-mean welfare maximizing allocation for all the other p-values.

Theorem 4. *When Ψ corresponds to the p-mean welfare objective with finite $p < 1$ and $p \neq 0$, Bivalued Yankee Swap run with the following gain function computes a p-mean welfare allocation.*

$$\phi_{p\text{-}\texttt{W}}(X, i, d) = \begin{cases} (v_i(X_i) + d)^p - v_i(X_i)^p & p \in (0,1) \text{ and } v_i(X_i) > 0 \\ v_i(X_i)^p - (v_i(X_i) + d)^p & p < 0 \text{ and } v_i(X_i) > 0 \\ Md & v_i(X_i) = 0 \end{cases}$$

where M is a very large number.

Note that Theorem 4 does not include the case where $p = 1$. This is because the gain function $\phi(X, i, d) = d$ does not satisfy (G3). The case where $p = 1$ corresponds to the utilitarian social welfare (`USW`) of an allocation. While we can construct a valid gain function to compute a `USW` optimal allocation, we do not need to. There exists an efficient algorithm for computing a `USW` optimal allocation without using Bivalued Yankee Swap; that is, compute a clean utilitarian welfare maximizing allocation with respect to the binary submodular valuations $\{\beta_i\}_{i \in N}$ and allocate the remaining goods arbitrarily.

5.4 When c is not a Natural Number

If c is not a natural number (or equivalently, a does not divide b), the complexity of the problem increases significantly and Bivalued Yankee Swap no longer works. This complexity is captured by [1], who show that computing MNW allocations under *bivalued additive valuations* when $a \in \mathbb{N}_{\geq 1}$ and $b \in \mathbb{N}_{\geq 1}$ are coprime is NP-hard. More formally, they show that for every coprime $a \geq 3$ and $b > a$, the problem of computing an `MNW` allocation is NP-hard. Note that this hardness result does not cover the case where $a = 2$; this case has been recently shown to be in P for additive valuations [2] but remains open for submodular valuations.

6 Maximin Share Guarantees of MNW and Leximin Allocations

We explore the maximin share guarantees of leximin and max Nash welfare allocations. The maximin share of an agent $i \in N$ (denoted by $\texttt{MMS}_i$) is defined as the utility agent i would receive if they divided the set of goods G into n bundles and picked the worst one (according to their preferences). More formally, $\texttt{MMS}_i = \max_X \min_{j \in N} v_i(X_j)$. An allocation X is defined as ε-$\texttt{MMS}$ for some $\varepsilon > 0$ if for all agents $i \in N$, $v_i(X_i) \geq \varepsilon \texttt{MMS}_i$ [11,27]. When agents have binary submodular valuations, both the max Nash welfare and the leximin allocation are guaranteed to be $\frac{1}{2}$-$\texttt{MMS}$. We prove the following two results about bivalued submodular valuations.

Theorem 5. *Let c be an integer ≥ 2. When agents have $\{1, c\}$-SUB valuations, then any max Nash welfare allocation X is $\frac{2}{5}$-$\texttt{MMS}$.*

Theorem 6. *Let c be an integer ≥ 2. When agents have $\{1, c\}$-SUB valuations, then any leximin allocation X is $\frac{1}{c+2}$-$\texttt{MMS}$.*

It is worth noting that the best known $\texttt{MMS}$-guarantee for submodular valuations is $\frac{1}{3}$ [21]. Theorem 5 shows that the max Nash welfare allocation offers better $\texttt{MMS}$ guarantees, albeit for a restricted subclass of submodular valuations.

7 Envy-Freeness of MNW and Leximin Allocations

Our final technical section deals with the envy-freeness of max Nash welfare and leximin allocations. An allocation X is *envy free up to one good (EF1)* if for all agents $i, j \in N$, $v_i(X_i) \geq v_i(X_j)$ or there exists a good $g \in X_j$ such that $v_i(X_i) \geq v_i(X_j - g)$ [11,24]. An allocation is *envy free up to any good (EFX)* if for all agents $i, j \in N$, and for all goods $g \in X_j$, we have $v_i(X_i) \geq v_i(X_j - g)$ [12].

Under binary submodular valuations, both the leximin and MNW allocations are known to be EFX [5]. Under bivalued additive valuations, MNW allocations are known to be EFX [3]. However, we now show that neither $\texttt{MNW}$ nor leximin allocations are EF1 under bivalued submodular valuations.

Proposition 1. *For every integer $c \geq 2$, neither the max Nash welfare nor the leximin allocation is guaranteed to be EF1 under $\{1, c\}$-SUB valuations.*

8 Conclusions and Future Work

In this work, we study fair allocation under bivalued submodular valuations. Our insights about this class of valuation functions enable us to use path transfers to compute allocations which satisfy strong fairness and efficiency guarantees.

As the first work to study bivalued submodular valuations, we believe our results are merely the tip of this iceberg. We believe that several other positive results can be shown, specifically with respect to `MMS` guarantees. It is unknown whether `MMS` allocations exist for this class of valuations. Prior results show that, while they may not exist under general submodular valuations [21], they indeed exist for the simpler classes of bivalued additive valuations [18] and binary submodular valuations [7]. Resolving this problem for bivalued submodular valuations is a natural next step.

We also believe extending these results and the technique of path transfers beyond bivalued submodular valuations is a worthy pursuit. It is unlikely that we will be able to compute optimal `MNW` or leximin allocations due to several known intractability results. However, we conjecture that it is possible to use a Yankee Swap based method to compute approximate max Nash welfare allocations for more general classes of submodular valuations. One specific class of interest is that of trivalued submodular valuations, e.g. the marginal gain of each good is either 0, 1 or $c > 1$. We intend to explore this class in future work.

Acknowledgements The authors would like to thank anonymous WINE 2023 reviewers for 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. 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: The half-integer case. CoRR **abs/2207.10949** (2022)
3. Amanatidis, G., Birmpas, G., Filos-Ratsikas, A., Hollender, A., Voudouris, A.A.: Maximum nash welfare and other stories about efx. Theoretical Computer Science **863**, 69–85 (2021)
4. Aziz, H., Li, B., Moulin, H., Wu, X.: Algorithmic fair allocation of indivisible items: A survey and new questions. CoRR **abs/2202.08713** (2022)
5. 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)
6. Barman, S., Bhaskar, U., Krishna, A., Sundaram, R.G.: Tight Approximation Algorithms for p-Mean Welfare Under Subadditive Valuations. In: Proceedings of the 28th Annual European Symposium on Algorithms (ESA). pp. 11:1–11:17 (2020)
7. 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)
8. Barman, S., Verma, P.: Truthful and fair mechanisms for matroid-rank valuations. In: Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI) (2022)

9. Benabbou, N., Chakraborty, M., Elkind, E., Zick, Y.: Fairness towards groups of agents in the allocation of indivisible items. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI). pp. 95–101 (2019)
10. 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)
11. Budish, E.: The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. Journal of Political Economy **119**(6), 1061 – 1103 (2011)
12. Caragiannis, I., Kurokawa, D., Moulin, H., Procaccia, A.D., Shah, N., Wang, J.: The unreasonable fairness of maximum nash welfare. In: Proceedings of the 17th ACM Conference on Economics and Computation (EC). p. 305–322 (2016)
13. Chakraborty, M., Igarashi, A., Suksompong, W., Zick, Y.: Weighted envy-freeness in indivisible item allocation. ACM Transactions on Economics and Computation **9** (2021)
14. Cousins, C.: An axiomatic theory of provably-fair welfare-centric machine learning. In: Proceedings of the 35th Annual Conference on Neural Information Processing Systems (NeurIPS). pp. 16610–16621 (2021)
15. Cousins, C.: Bounds and Applications of Concentration of Measure in Fair Machine Learning and Data Science. Ph.D. thesis, Brown University (2021)
16. Cousins, C.: Uncertainty and the social planner's problem: Why sample complexity matters. In: Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency (2022)
17. Cousins, C.: Revisiting fair-PAC learning and the axioms of cardinal welfare. In: Artificial Intelligence and Statistics (AISTATS) (2023)
18. 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)
19. 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)
20. 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)
21. Ghodsi, M., HajiAghayi, M., Seddighin, M., Seddighin, S., Yami, H.: Fair allocation of indivisible goods: Improvements and generalizations. In: Proceedings of the 19th ACM Conference on Economics and Computation (EC). pp. 539–556 (2018)
22. Heidari, H., Ferrari, C., Gummadi, K., Krause, A.: Fairness behind a veil of ignorance: A welfare analysis for automated decision making. In: Advances in Neural Information Processing Systems. pp. 1265–1276 (2018)
23. Kurokawa, D., Procaccia, A.D., Shah, N.: Leximin allocations in the real world. ACM Transactions of Economics and Computation **6**(3–4) (2018)
24. 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)
25. Moulin, H.: Fair division and collective welfare. MIT Press (2004)
26. Plaut, B., Roughgarden, T.: Almost envy-freeness with general valuations. SIAM Journal on Discrete Mathematics **34**(2), 1039–1068 (2020)
27. 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)

28. Viswanathan, V., Zick, Y.: A general framework for fair allocation with matroid rank valuations. In: Proceedings of the 24th ACM Conference on Economics and Computation (EC) (2023)
29. 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)