On the Hardness of Fair Allocation under Ternary Valuations

Zack Fitzsimmons
College of the Holy Cross
Worcester, MA, United States
zfitzsim@holycross.edu

Vignesh Viswanathan
University of Massachusetts
Amherst, MA, United States
vviswanathan@umass.edu

Yair Zick
University of Massachusetts
Amherst, MA, United States
yzick@umass.edu

ABSTRACT

We study the problem of fair allocation of indivisible items when agents have ternary additive valuations — each agent values each item at some fixed integer values a, b, or c that are common to all agents. The notions of fairness we consider are max Nash welfare (MNW), when a, b, and c are non-negative, and max egalitarian welfare (MEW). We show that for *any* distinct non-negative a, b, and c, maximizing Nash welfare is APX-hard — i.e., the problem does not admit a PTAS unless P = NP. We also show that for any distinct a, b, and c, maximizing egalitarian welfare is APX-hard except for a few cases when $b = 0$ that admit efficient algorithms. These results make significant progress towards completely characterizing the complexity of computing exact MNW allocations and MEW allocations. En route, we resolve open questions left by prior work regarding the complexity of computing MNW allocations under bivalued valuations, and MEW allocations under ternary mixed manna.

KEYWORDS

Fair Allocation; Nash Welfare; APX Hardness

ACM Reference Format:
Zack Fitzsimmons, Vignesh Viswanathan, and Yair Zick. 2025. On the Hardness of Fair Allocation under Ternary Valuations. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Fair allocation of indivisible items is a fundamental problem in computational social choice. We are given a set of indivisible *items* that need to be distributed among *agents* that have subjective *valuations* for the items they receive. Many problems can be naturally cast as instances of the fair allocation problem. For example, one might wish to distribute a set of course seats to students, or schedule shifts to hospital workers. Our objective is to find an *allocation* of items to agents satisfying certain natural *justice criteria*. Unfortunately, when agents have arbitrary combinatorial valuations, several allocation desiderata are either computationally intractable to compute, or simply not guaranteed to exist (see e.g., Caragiannis et al. [15], Plaut and Roughgarden [38] as well as Amanatidis et al. [3] for an overview).

Thus, recent works study simpler classes of valuations where exact fair allocations can be computed. There is an efficient algorithm to compute Max Nash welfare allocations when agents have binary valuations [7, 9, 31], i.e., where each item is valued at either 0 or 1. This result was later extended to bivalued additive valuations where each item is valued at 1 or c with c being either an integer or a half-integer [1, 2]. Similarly, for the problem of allocating chores, there exists an efficient algorithm that computes leximin allocations when agents have binary costs [11] or bivalued costs when the ratio of the costs is 2 [34].

Other works restrict their attention to bivalued instances in the realm of goods [4, 28] as well as chores [6, 17, 23, 29]. Generalizing beyond bivalued instances, much less is known about the complexity of fair allocation under ternary or trivalued instances: when each item is valued at a, b, or c for some integers a, b, and c. For example, we do not know if an exact max Nash welfare allocation is efficiently computable when each item is valued at 0, 1, or 2. Our goal in this paper is to bridge this gap by answering the following question:

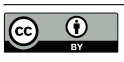
> *What is the computational complexity of computing fair allocations under ternary valuations?*

1.1 Our Results

The question has been partially answered in the literature. Garg et al. [25] (and Amanatidis et al. [4]) show that when each item is valued at 0, 1, or c, computing a max Nash welfare allocation is APX-hard with a large enough c. The hardness results for computing max Nash welfare allocations under bivalued valuations [1, 2] also extends to some classes of trivalued valuations. Building upon these results, we offer a comprehensive analysis of the complexity of computing fair allocations when agents have ternary or trivalued valuations, i.e., items are valued at arbitrary integers a, b, or c ($a < b < c$). A summary of our results is presented in Table 1.

We study the objectives of maximizing Nash welfare and maximizing egalitarian welfare (also known as the Santa Claus objective [8]). These are two of the most popular notions of fairness in the literature, and are extremely well studied. The Nash welfare of an allocation is defined as the product of agent utilities, and the egalitarian welfare of an allocation is defined as the utility of the worst-off agent.

We first study the *all goods* setting; here, items have a non-negative marginal value for agents. We show that the problems of computing a max Nash welfare and a max egalitarian welfare allocation are APX-hard for any a, b, c such that $0 \leq a < b < c$ (Theorems 3.1 and 3.5). This result completely characterizes the complexity of computing max Nash welfare allocations when agents have ternary valuations. Importantly, this result shows that even when agents have $\{0, 1, 2\}$ valuations, computing a max Nash welfare allocation is hard. A similar result almost completely characterizes the complexity of maximizing Nash welfare under bivalued valuations [1, 2]; these results, however, do not resolve the APX-hardness of the problem for the specific case when one of the values an item



Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

Valuation Class	Additive
$\{a, b\}$ $a = 0, 1$ or 2	*Poly* [1, 2, 9]
$\{a, b\}$ $b > a > 3$	APX-hard [1]
$\{a, b\}$ $b > a = 3$	NP-hard [1] **APX-hard** (Proposition 3.8)
$\{0, 1, c\}$ some large c	APX-hard [25]
$\{a, b, c\}$ $c > b > a \geq 0$	**APX-hard** (Theorems 3.1 and 3.5)

(a) Complexity of computing max Nash welfare

Valuation Class	Additive	Submodular
$\{-1, 0\}$	trivial	*Poly* [11]
$\{-1, 0, 1\}$	*Poly* [21]	**NP-hard** (Theorem 4.7)
$\{-1, 0, c\}$ $c > 1$	*Poly* [21]	**Poly** (Proposition 4.8)
$\{-2, 0, c\}$ $c \geq 3$	open	open
$\{a, b, c\}$ all other cases	**NP-hard** (Theorems 4.1, 4.3 and [21])	**NP-hard** (Theorems 4.1, 4.3 and [21])

(b) Complexity of computing max egalitarian welfare

Table 1: Summary of our results. Here, a, b, and c denote arbitrary distinct co-prime integers. Our contributions are highlighted in bold. By valuation class $\{a, b, c\}$, we mean instances where every item is valued at either a, b, or c by all the agents.

can have is 3.[1] We resolve this case as well, showing APX-hardness and completing their characterization under bivalued valuations (Proposition 3.8).

Next, we study the *mixed manna* setting, where items can have both positive and negative marginal values. For the special case when $a = -1$, $b = 0$, and c is an arbitrary integer, an efficient algorithm is known to compute max egalitarian welfare allocations [21]. We show that generalizing beyond this case is unlikely by showing that computing a max egalitarian welfare allocation is NP-hard for almost every other a, b, and c.

In line with the questions posed by [7] and [20], we also ask the question of whether the results of Cousins et al. [21] can be generalized to submodular valuations. We find that apart from the case where $c = 1$, their results can in fact be generalized to submodular valuations (Proposition 4.8). Somewhat surprisingly, for the special case where $a = -1, b = 0$, and $c = 1$, the problem of computing a max egalitarian welfare allocation is NP-hard when agents have submodular valuations (Theorem 4.7).

Due to space constraints, several proofs have been omitted. The complete set of proofs can be found in the full version of the paper [24].

1.2 Additional Related Work

Current known results on trivalued valuations consider special cases, e.g., algorithms when $b = 0$ [21] or hardness when c is much larger than a and b [4, 25].

When all the items are *chores*, the complexity of computing egalitarian allocations is resolved by prior work [17, 34]. It is known that unless each item is valued at one of two values a, b ($a < b$) such that $a = 2b$, the problem is APX-hard. The case where $a = 2b$ admits a polynomial time algorithm. This implies that for all ternary values, unless $a = -2, b = -1$ and $c = 0$, the problem is APX-hard. Note that this special case can be solved similar to the case where $a = 2b$, since any items valued at 0 by one of the agents can be allocated to them without affecting the egalitarian welfare.

[1]For every other case, Akrami et al. [1] show that the problem is either APX-hard or admits an efficient algorithm.

Aside from exact algorithms, a long line of fascinating work studies approximation algorithms for maximizing Nash welfare [10, 19, 22, 26, 27, 35] and egalitarian welfare [5, 8, 16, 34]. The current best known approximation ratios for maximizing Nash welfare are 1.45 for additive valuations [10] and $4+\epsilon$ for submodular valuations [26]. There is also a constant factor algorithm under subadditive valuations which uses a polynomial number of demand queries [22].

There is a 0.5-approximation algorithm for maximizing egalitarian welfare when all items are chores [34], but there is no constant approximation for the all goods case. However, the special case of the Santa Claus problem admits a 13-approximation algorithm [5].

Lee [33] show APX-hardness of the Max Nash welfare problem under general additive valuations, Akrami et al. [1] show APX-hardness for some cases when agents have bivalued additive valuations, and Garg et al. [25] show APX-hardness when agents have $\{0, 1, c\}$ valuations for some large constant c.

2 PRELIMINARIES

For any $k \in \mathbb{N}$, we use $[k]$ to denote the set $\{1, 2, \ldots, k\}$. We have a set of n *agents* $N = [n]$ and m *items* $O = \{o_1, \ldots, o_m\}$. Each agent $i \in N$ has a *valuation function* $v_i : 2^O \to \mathbb{R}$; $v_i(S)$ specifies agent i's utility for the bundle of items S. We primarily assume *additive* valuations: a valuation function is additive if for all $S \subseteq O$, $v_i(S) = \sum_{o \in O} v_i(\{o\})$. For readability, we sometimes abuse notation and use $v_i(o)$ to denote $v_i(\{o\})$.

We often assume that agents have restricted values for items. More formally, given a set $A \subseteq \mathbb{Z}$, agent i has A-valuations if v_i is additive and $v_i(o) \in A$ for all $o \in O$. Specifically, we often consider $\{a, b, c\}$-valuations for some integers a, b, and c. Throughout the paper, a, b, and c will only be used to denote integers. An *allocation* $X = (X_1, \ldots, X_n)$ is an n-partition of the set of items O, where agent i receives the bundle X_i. We require that every item is allocated to some agent. This constraint is required when items can be negatively valued by agents. The *utility* of agent i under the allocation X is $v_i(X_i)$.

2.1 Fairness Notions (or Objectives)

We consider two fairness objectives in this paper.

Max Nash Welfare (MNW): the Nash welfare of an allocation X is defined as the geometric mean of agent utilities $\text{NSW}(v, X) = \left(\prod_{i \in N} v_i(X_i)\right)^{1/n}$.

Max Egalitarian Welfare (MEW): The egalitarian welfare of an allocation X is defined as the minimum utility obtained by any agent in the allocation i.e. $\min_{i \in N} v_i(X_i)$. An allocation which maximizes this objective value is referred to as a max egalitarian welfare allocation.

Since the max Nash welfare objective makes little sense when some agents have negative utilities, we only study it when all agents have non-negative utilities. Some of our proofs will also use the utilitarian social welfare of an allocation to establish some bounds on the Nash (or egalitarian) welfare. The *utilitarian social welfare* of an allocation X is defined as the sum of agent utilities $\sum_{i \in N} v_i(X_i)$.

2.2 Approximation Algorithms and APX-hardness

For some $\alpha > 1$, an algorithm is an α-approximation algorithm for the max Nash welfare objective if it always outputs an allocation which has Nash welfare at least $\frac{1}{\alpha}$ of the optimal value. We similarly define an α-approximation algorithm for the max egalitarian welfare.

Approximation algorithms are usually only defined when the objective value is either always positive or always negative. This is true of our fairness objectives in the all goods case. However, in the mixed goods and chores case, where items can have both positive and negative values, the optimal egalitarian welfare could be positive while many allocations could have a negative egalitarian welfare. Thus, in that case, we do not discuss approximability and only discuss NP-hardness and exact algorithms.

For most valuation classes, we show the hardness of computing fair allocations by proving APX-hardness [36]. APX-hard problems do not admit a Polynomial Time Approximation Scheme (PTAS) unless P = NP. This is equivalent to saying that there exists an $\alpha > 1$ such that the problem does not admit an α-approximation algorithm unless P = NP.

The class APX consists of all the problems which admit efficient constant factor approximation schemes. As mentioned in the related work section, existing results put the problem of maximizing Nash welfare in APX for general additive valuations [10, 19]. These results show that our constant factor lower bounds, are in some sense tight. That is, it would be impossible to improve these lower bounds to $\Omega(\log n)$ or $\Omega(poly(n))$; the tightest possible lower bounds are constant.

3 THE ALL GOODS CASE: $0 \leq a < b < c$

We first consider the case where $0 \leq a < b < c$ and all agents have $\{a, b, c\}$-valuations. It is known that MNW allocations can be computed efficiently when agents have $\{0, 1\}$-valuations [9], $\{1, c\}$-valuations with $c > 1$ [1], and $\{2, c\}$-valuations with c odd and at least 3 [2]. MEW allocations can be computed efficiently when agents have $\{0, 1\}$-valuations [7, 31], and $\{1, c\}$-valuations with $c > 1$ [1, 20].

However, the complexity of the $\{0, 1, 2\}$ case is unknown. Our first result resolves this.

Theorem 3.1. *When agents have $\{a, b, c\}$-valuations with $0 \leq a < b < c$ and $c \leq 2b$, computing an MNW allocation is* APX*-hard.*

The proof of this result is highly involved and uses a careful reduction from a variant of 3SAT. To provide a high level idea of the proof, we instead present a simpler result, and discuss how it can be generalized.

Theorem 3.2. *When agents have $\{0, 1, 2\}$-valuations, computing an MNW allocation is* APX*-hard.*

Proof. We show APX-hardness by using the Max-2P2N-3SAT problem[2]:

Name: Max-2P2N-3SAT

Given: A boolean formula $\phi(x_1, \ldots, x_n)$ where ϕ is in 3CNF and each variable x_i occurs in ϕ exactly twice as positive literal and twice as a negated literal.

Question: Find the assignment σ that maximizes the total number of clauses satisfied in ϕ.

We specifically use the following lemma which proves APX-hardness of the Max-2P2N-3SAT problem.

Lemma 3.3 ([14]). *Given an instance of Max-2P2N-3SAT and any $\epsilon > 0$, it is* NP*-hard to decide if $(1 - \epsilon)$ fraction of the clauses can be satisfied or if all solutions satisfy at most a $\frac{1015}{1016} + \epsilon$ fraction of the clauses.*

Given an instance $\phi(x_1, \ldots, x_n)$ of Max-2P2N-3SAT with $m = 4n/3$ clauses[3] $C_1, \ldots, C_m$, we construct an instance of an allocation problem with $7n$ items and $4n$ agents.

For each variable x_i, we create 5 items: x_i, x_i' corresponding to the positive literal, $\overline{x}_i, \overline{x}_i'$ corresponding to the negative literal, and a *clog* item $clog_i$. We also add $2n$ special items $d_1, \ldots, d_{2n}$.

Our instance will have a set of $4n$ agents with the following valuations. For each variable x_i, we create an agent pos_i who values x_i, x_i' at 1 and $clog_i$ at 2. Similarly, for each variable x_i, we create an agent neg_i who values $\overline{x}_i, \overline{x}_i'$ at 1 and $clog_i$ at 2. For each clause C_i, we create an agent C_i who values the items corresponding to the literals in C_i at 1 and the special item d_i at 1. We have a set of $2n - m$ dummy agents: $s_1, \ldots, s_{2n-m}$ where for each $i, 1 \leq i \leq 2n - m$, s_i values all literal items and the special item d_{m+i} at 1. All unmentioned valuations are at 0. See Figure 1 for an example of this construction.

By our construction, each special item is valued positively by exactly one agent. We now prove correctness of our reduction.

($\Longrightarrow$) Suppose there is an assignment $\sigma : \{x_1, \ldots, x_n\} \to \{0, 1\}$ to the variables $x_1, \ldots, x_n$ that satisfies at least $(1 - \epsilon)$ of the clauses in ϕ. We construct an allocation X using this assignment.

We first allocate the items to the variable agents. For each $i, 1 \leq i \leq n$, if $\sigma(x_i) = 1$ we allocate $clog_i$ to pos_i and $\overline{x}_i, \overline{x}_i'$ to neg_i;

[2] We mention here that our APX-hardness reduction shares some of the general structure with the NP-hardness proof for computing a MNW allocation for $\{0, 1, c\}$-valuations when c is unboundedly large [4]. However, to handle the case of fixed c and to generalize to APX-hardness, our construction requires a different structure to the valuations and additional padding agents/items.

[3] This value of m comes from the fact that each clause has 3 literals, and each variable appears exactly twice as a positive literal and twice as a negative literal.

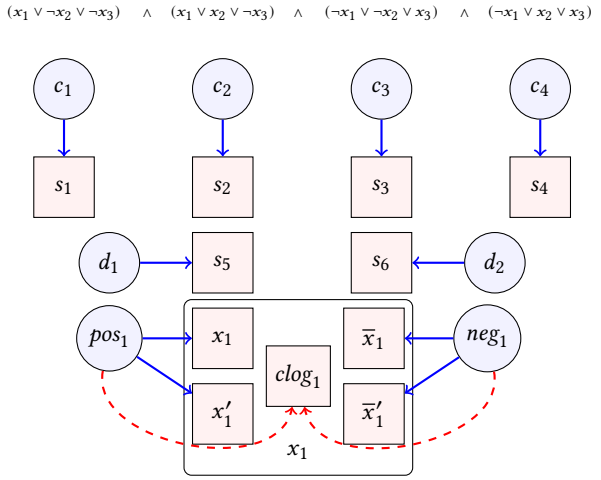


Figure 1: An illustration of the reduction from an instance of 2P2N-3SAT to the MNW problem. Blue circles represent agents, and red squares represent items. The j-th clause in the formula induces a corresponding clause agent c_j who likes all items generated by its literals, plus the special item s_j. Each variable (here, we just present the elements generated by x_1) produces five items and two agents. Agent pos_1 values x_1 and x'_1 at 1, and $clog_1$ at 2. Agent neg_1 values $\overline{x}_1$ and $\overline{x}'_1$ at 1, and $clog_1$ at 2. The dummy agent d_1 likes s_5 and the dummy agent d_2 likes s_6, and the items generated by literals at 1.

if $\sigma(x_i) = 0$ we allocate $clog_i$ to neg_i and x_i, x'_i to pos_i. Thus, if $\sigma(x_i) = 1$, the agent pos_i gets the clog item for a utility of 2, and the agent neg_i gets a utility of 2 from the two literal items assigned to it.

For each clause C_i that is satisfied by the assignment σ, we allocate exactly one copy of one of the literal items that satisfies that clause to the corresponding clause agent. For example, if $C_i = (\overline{x}_1 \vee x_2 \vee x_3)$ and $\sigma(x_1) = 0, \sigma(x_2) = 0$, and $\sigma(x_3) = 1$ then we can allocate $\overline{x}_1, \overline{x}'_1, x_3$ or x'_3 to the clause agent C_i; the exact choice can be made arbitrarily. Thus, if C_i is a satisfied clause, we allocate one literal item to its corresponding agent for a utility of 1. Each clause agent also uniquely values a special item at 1, and we allocate this item to them. Overall, each clause agent corresponding to a satisfied clause receives a utility of 2 and each clause agent corresponding to an unsatisfied clause receives only their special item for a utility of 1.

For each of the dummy agents, we allocate one of the remaining literal items along with the unique special item they value at 1. There are at least $2n - m$ such literal items remaining so this is possible.

At the end of this process, there are at most ϵm unallocated literal items. With small enough ϵ, this is less than $2n - m = 0.5m$. So we allocate the remaining items among the dummy agents such that no agent gets more than one such item.

Let us take stock of the utility of our agents at this stage. There are:

- $2n$ literal agents who received either two literal items or one clog item. So they get a utility of 2.
- $m - m'$ satisfied clause agents who receive one literal item and their corresponding special item for a utility of 2.
- $m' (\leq \epsilon m)$ unsatisfied clause agents who receive their corresponding special item (and no literal item) for a utility of 1.
- $2n - m - m'$ dummy agents who receive a literal item which they value at 1, and their corresponding special item which they value at 1, for a total utility of 2.
- m' dummy agents who receive their corresponding special item which they value at 1 and two literal items for a total utility of 3.

In the above accounting, we use m' to denote the number of unsatisfied clauses. We have the following NSW for our allocation.

$$\text{NSW}(v, X) = \left(\prod_{i \in N} v_i(X_i)\right)^{\frac{1}{4n}} = \left(2^{4n-2m'} 3^{m'}\right)^{\frac{1}{4n}} \geq 2\left(\frac{3}{4}\right)^{\epsilon/3} \quad (1)$$

The final inequality holds since $m = 4n/3$ and $m' \leq \epsilon m$.

($\Longleftarrow$) For the other direction, we assume that assignments to ϕ satisfy at most $(\frac{1015}{1016} + \epsilon)m$ clauses, and upper bound the max Nash welfare. Consider an arbitrary max Nash welfare allocation X. To upper bound the Nash welfare of X, we prove some important properties about X. These properties can be assumed without loss of generality; that is, these properties are satisfied by at least one max Nash welfare allocation X. At a high level, these properties show that X must be as egalitarian as possible subject to giving each item to an agent who values it positively.

Property 1. *In any MNW allocation X, all agents receive at least one item that gives them a positive utility.*

Proof. If the allocation does not do this the Nash welfare is 0. However, it is easy to find an allocation with positive Nash welfare. The non-literal agents can get their special item and the literal agents can share their corresponding clog and literal items so that they get a positive utility. □

Property 2. *In any MNW allocation X, all items are allocated to an agent who values the item positively.*

Proof. If an item is allocated to an agent who values it at 0, moving it to an agent who values it positively strictly Pareto dominates the allocation X, and therefore, improves the Nash welfare. □

At this point, we have fixed the allocation of all the $2n$ special items and the n clog items in X: special items go to the agents who value them at 1, and clog items go to the literal agents. This leaves us only with the $4n$ literal items.

For the rest of this proof, we assume X is a max Nash welfare allocation.

Property 3. *For each $i \in [n]$,*

(a) if $clog_i \in X_{pos_i}$, then $|X_{neg_i}| = 2$.
(b) if $clog_i \in X_{neg_i}$, then $|X_{pos_i}| = 2$.

Proof. We only show (a); an analogous argument holds for (b). We assume that the literal agent pos_i has the clog item $clog_i$. Of the remaining items, neg_i has exactly two items they value at 1: the items $\overline{x}_i$ and $\overline{x}'_i$. neg_i must get at least one of these items (say $\overline{x}'_i$); otherwise they will receive a utility of 0, violating Property 1. If they do not receive x_i as well, it must be allocated to either a clause agent or a dummy agent (Property 2). Let this agent be z. Since z is a clause or dummy agent, they receive their corresponding special item as well (Property 2); this means they have a utility of at least 2. Since the agent neg_i has a utility of exactly 1, moving the item x_i from a to neg_i results in a weak improvement in Nash welfare. □

For the next property, we derive a truth assignment σ for the 2P2N-3SAT instance ϕ from the allocation X. If $clog_i$ is allocated to pos_i then $\sigma(x_i) = 1$ otherwise $\sigma(x_i) = 0$. Recall that all assignments to ϕ satisfy at most $\left(\frac{1015}{1016} + \epsilon\right)$ fraction of the clauses, and so σ satisfies some, but not all, of the clauses.

Property 4. *In X, any clause agent C_i that is not satisfied by the assignment σ receives a utility of* 1.

Proof. From Property 3, all the items that correspond to literals with sign opposite to the assignment get allocated to the literal agents. That is, if $\sigma(x_i) = 1$, then items $\overline{x}_i$ and $\overline{x}'_i$ are allocated to neg_i.

Any clause agent C_i receives their corresponding special item (Property 2). This gives the clause agent a utility of at least one. We show that any clause agent C_i that is not satisfied by the assignment σ does not receive any other item.

This follows from the fact that all the items which are positively valued by C_i correspond to literal items with sign opposite to the assignment. All such literal items are allocated to literal agents. Therefore, any unsatisfied clause agent C_i receives exactly one item. □

Assume there are m' unsatisfied clauses in the instance. From Property 4, at least m' clause agents receive a utility of exactly 1. The maximum utilitarian social welfare possible in our constructed instance is $8n$. Conditioned on at least m' agents receiving a utility of 1, the max Nash welfare possible occurs when the remaining possible utility is divided equally among the other agents. That is, the remaining $4n - m'$ agents receive a utility of $\frac{8n-m'}{4n-m'}$. This may not be possible to achieve but serves as an upper bound on the Nash welfare of any allocation in this instance. We can now bound the Nash welfare of any allocation X in this instance as:

$$\begin{aligned}\mathrm{NSW}(v, X) = \left(\prod_{i \in N} v_i(X_i)\right)^{\frac{1}{4n}} &\leq \left(\frac{8n - m'}{4n - m'}\right)^{\frac{4n-m'}{4n}} \\ &\leq \left[\frac{2 - \left(\frac{1}{1016} - \epsilon\right)\frac{1}{3}}{1 - \left(\frac{1}{1016} - \epsilon\right)\frac{1}{3}}\right]^{1-\left(\frac{1}{1016}-\epsilon\right)\frac{1}{3}}. \end{aligned} \tag{2}$$

The last inequality follows from using

$$m' \geq \left(\frac{1}{1016} - \epsilon\right) m = \left(\frac{1}{1016} - \epsilon\right) \frac{4n}{3}.$$

What we have shown is that it is NP-hard to decide if an instance has max Nash welfare at least (1), or if all allocations have Nash welfare at most (2). Dividing these two values gives us a lower bound on how well we can approximate Nash welfare. Setting $\epsilon = 10^{-6}$, the approximation lower bound this gives us is 1.00006. □

To generalize to $\{a, b, c\}$ valuations we retain the same high level reduction but the construction is more complex to account for the different ways agents can receive utility. Specifically, we must account for the fact that agents could receive items at value a when $a > 0$. We take care of this case by utilizing different kinds of local transfers (similar to the argument in Property 3) to give allocations more structure.

Recently, Jain and Vaish [32] studied the problem of maximizing Nash welfare under two sided preferences and show that the problem is NP-hard under $\{0, 1, 2\}$ valuations and capacity constraints. Since fair allocation is a special case of their problem, Theorem 3.1 presents an improved hardness result for their problem since we show APX-hardness and eliminate the need for capacity constraints.

Our next result resolves the case when $2b < c$. It may be possible to use a similar 2P2N-3SAT construction in this case as well but our proof uses a much simpler vertex cover based reduction. Specifically, we reduce from the following problem.

Lemma 3.4. *There exists a constant $\gamma \in (0, 1)$ such that, given a 3-regular graph G and an integer k, it is* NP*-hard to decide if G has a vertex cover of size k that covers all edges or all subsets of nodes of size k cover at most a $(1 - \gamma)$ fraction of the edges of G.*

This follows from applying the arguments of Petrank [37] to the min vertex cover hardness result of Chlebík and Chlebíková [18].

Theorem 3.5. *When agents have $\{a, b, c\}$-valuations with $0 \leq a < b < c$ and $2b < c$, computing an MNW allocation is* APX*-hard.*

Proof. We reduce from Lemma 3.4. We are given a 3-regular graph $G = (V, E)$ and an integer k. Note that since the graph is 3-regular, $|E| = \frac{3|V|}{2}$, and k must be at least $\frac{|V|}{2}$ for this problem to be non-trivial. This is because we need at least $\frac{|V|}{2}$ nodes to cover $\frac{3|V|}{2}$ edges in a 3-regular graph.

We construct a fair allocation instance with $3k - 0.5|V|$ agents and $7k - 1.5|V|$ items.

The $7k - 1.5|V|$ items are defined as follows:

(a) For each each edge $e_{ij} \in E$ we have an item e_{ij},
(b) We have k *vertex cover* items $c_1, \ldots, c_k$, and
(c) We have $6k - 3|V|$ special items.

The $3k - 0.5|V|$ agents have valuations defined as follows:

Node Agents: For each node $i \in V$, we have an agent who values the edges incident on it at b and

Dummies: We have $3k - 1.5|V|$ dummy agents, who value edge items at b, and exactly two special items each at b. Since there are $6k - 3|V|$ special items, we can ensure that no two dummy agents value the same special item at b.

The vertex cover items are valued at c by both the node and dummy agents. All unmentioned values are at a.

We now prove correctness of our reduction.

Assume the graph admits a vertex cover (say S) of size k. We allocate the k vertex cover items to the agents in S. All other node

agents receive the edge items corresponding to the three edges they are incident on. This is feasible since at least one endpoint of each edge belongs to the vertex cover S.

At this point, we only have to allocate the special items, and perhaps some edge items (if both endpoints of some edge belong to the vertex cover S). We allocate the remaining items to the dummy agents. Each dummy agent receives their two special items and a single unassigned edge item; this ensures all $7k - 1.5|V|$ items are assigned.

The k agents in the vertex cover S have a utility of c, and the remaining $|V| - k$ node agents have a utility of $3b$. The $3k - 1.5|V|$ dummy agents have a utility of $3b$ as well. Thus, the Nash welfare of this allocation is

$$\text{NSW}(v, X) = \left(c^k (3b)^{(2k-0.5|V|)}\right)^{\frac{1}{3k-0.5|V|}} \qquad (\text{NSW}^+)$$

For the other direction, assume that no subset of nodes of size k covers more than $(1-\gamma)|E|$ edges. For this case, we slightly tweak the valuation function to make our analysis easier. Specifically, we change the valuation functions such that all valuations at a are replaced with a' such that $a' = \max\{a, \frac{2b^2}{c}, \frac{2b}{3}\}$. a' may no longer be an integer but it is guaranteed to be less than b since $2b < c$. We refer to this new valuation profile using v'. Crucially, since we only increased agent valuations, we must have, for any allocation X, $\text{NSW}(v, X) \leq \text{NSW}(v', X)$. Thus, any upper bound on the max Nash welfare under the valuations v' also bounds the max Nash welfare under the valuations v. To upper bound the Nash welfare, we examine the MNW allocation X.

Lemma 3.6. *There exists an MNW allocation X with the following properties.*

(1) No agent receives more than one vertex cover item.
(2) An agent who receives a vertex cover item does not receive any other item in X.
(3) No agent receives four or more items.

Proof. We separately consider each property of X stated in the lemma.

(1) If an agent has two vertex cover items, there must be some agent j who does not receive a vertex cover item and receives at most two items. Moving one of the vertex cover items to agent j's bundle strictly improves Nash welfare. This follows from the fact that $c > 2b \geq v'_j(X_j)$.
(2) If an agent i with a vertex cover item c_r has another item (say o), then there must be an agent j without a vertex cover item that has at most two items. Transferring o to agent j weakly improves the Nash welfare. This is because

$$\frac{v'_i(X_i) - v'_i(o)}{v'_i(X_i)} \geq \frac{c}{b+c} \geq \frac{2b}{a'+2b} \geq \frac{v'_j(X_j)}{v'_j(X_j) + v'_j(o)}$$

The second inequality follows by plugging in $c \geq \frac{2b^2}{a'}$.
(3) If an agent has four or more items, we can transfer the least valued item (out of the four or more) to an agent who receives at most two items and no vertex cover items. Crucially, this uses the fact that $a' \geq \frac{2b}{3}$.

Note that the transfers to show properties 2 and 3 only weakly improve Nash welfare, but these transfers are made without violating the other properties; additionally, these transfers need to be made only a finite number of times to ensure the property holds. So there must exist an MNW allocation where all three properties are satisfied. □

The three properties stated in Lemma 3.6 offer us some structure. Each agent in the max Nash welfare allocation X either has a vertex cover item or exactly three other items.

Next, we lower bound the number of agents who receive three items but do not receive a utility of $3b$. Consider the subset of nodes consisting of the agents who receive a vertex cover item. This subset of nodes, by assumption, must *not* cover at least $\gamma|E|$ edges. Each of the uncovered edges represents an edge item that both endpoints value at b, but can only be given to one of them. Thus, one of the uncovered edge's endpoints must receive a utility of less than $3b$. If we do this for all uncovered edges, we get that there are at least $\frac{\gamma|E|}{3}$ agents whose utility is less than $3b$. We divide by three since G is 3-regular, so each node is counted at most thrice.

These $\frac{\gamma|E|}{3}$ agents receive a utility of at most $2b + a'$. All other agents receive a utility of either $3b$ or c. Note that receiving a utility of more than $3b$ is impossible with only three items, as the only items valued at c are the vertex cover items. This upper bounds the Nash welfare of the allocation X under v'. Assuming m' agents do not receive a vertex cover item or a utility of $3b$, $\text{NSW}(v', X)$ is upper bounded by

$$\left(c^k (3b)^{2k-0.5|V|-m'} (2b+a')^{m'}\right)^{\frac{1}{3k-0.5|V|}}$$
$$\leq \left(c^k (3b)^{2k-0.5|V|} \left(\frac{2b+a'}{3b}\right)^{\frac{\gamma|E|}{3}}\right)^{\frac{1}{3k-0.5|V|}} \qquad (\text{NSW}^-)$$

The inequality holds since $m' \geq \frac{\gamma|E|}{3}$. We have shown that it is NP-hard to decide whether an allocation has Nash welfare at least NSW^+ or whether all allocations have a Nash welfare of at most NSW^-. Taking the ratio of the two gives us the following approximation lower bound

$$\frac{\text{NSW}^+}{\text{NSW}^-} = \left(\left(\frac{3b}{2b+a'}\right)^{\frac{\gamma|E|}{3}}\right)^{\frac{1}{3k-0.5|V|}} \geq \left(\frac{3b}{2b+a'}\right)^{\frac{\gamma}{5}}$$

The final inequality follows since $k \leq |V|$ and $|E| = 3|V|/2$. Since $a' < b$, this is a positive constant, and we are done. □

Lee [33] shows APX-hardness of the MNW problem for general additive valuations using the same min vertex cover problem [18], but their reduced instance is more general and not restricted to three fixed values a, b, and c. It is also worth noting that our proof leads to a constant factor lower bound of 1.00013 which improves on their constant factor lower bound of 1.00008.

Corollary 3.7. *Assume agents have $\{0, 1, 3\}$-valuations. It is impossible to approximate MNW by a factor smaller than* 1.00013 *unless* P = NP.

The vertex cover based proof technique also shows that computing max Nash welfare allocations is APX-hard even when agents have $\{3, c\}$ valuations where $c > 3$ and is not divisible by 3; this resolves an open question posed by Akrami et al. [1].

Proposition 3.8. *When agents have $\{3, c\}$-valuations with $c > 3$ and c not divisible by 3, computing an MNW allocation is* APX*-hard.*

We also show that the techniques used in this section can also be used to show APX-hardness for computing MEW allocations, when all items have non-negative value.

Theorem 3.9. *When agents have $\{a, b, c\}$-valuations with $0 \leq a < b < c$, computing an MEW allocation is* APX*-hard.*

Note that MEW allocations can be defined even when agents have negative values for the items; the above proof can be used to show NP-hardness for computing an MEW allocation even when a is negative.

4 MIXED MANNA

Next, we consider mixed manna, i.e. the case where agents have $\{a, b, c\}$-valuations with $a < b \leq 0 < c$. Note that when two of a, b and c are positive, the problem of computing MEW allocations is NP-hard (Theorem 3.9). For the cases when a and b are negative, we have the following hardness results.

Theorem 4.1. *When agents have $\{a, c\}$-valuations with $a < 0 < c$ and $|a| > |c|$, computing an MEW allocation is* NP*-hard.*

The above result follows from reductions using the decision version of the 2P2N-3SAT problem. We also have the following hardness result from Cousins et al. [21].

Theorem 4.2 ([21]). *When agents have $\{a, c\}$-valuations with (i) $a < 0 < c$, (ii) $|a| \geq 3$, and (iii) $|a|$ and $|c|$ are coprime, computing an MEW allocation is* NP*-hard.*

We can use these results to show the following, again using the 2P2N-3SAT problem:

Theorem 4.3. *When agents have $\{a, b, c\}$-valuations with $a < b < 0 < c$, computing an MEW allocation is* NP*-hard.*

Combining Theorems 4.1 to 4.3 shows that the only case where one could hope to compute an MEW allocation is when agents have $\{-2, 0, c\}$-valuations. We could not show a hardness result for this case and conjecture that it may admit efficient algorithms.

Conjecture 4.4. *There exists an efficient algorithm for computing MEW allocations when agents have $\{-2, 0, c\}$-valuations.*

For this case of $\{-1, 0, c\}$-valuations (with $c > 1$), an efficient algorithm to compute MEW allocations is known [21]. The algorithmic results of Cousins et al. [21] apply to a broader class of valuations they define as *order-neutral submodular valuations*. This is a more general class than additive valuations but a strict subset of submodular valuations. This restriction raises the natural question of whether the restriction from submodular valuations to order-neutral submodular valuations is necessary. The answer to this question turns out to be quite surprising. However, before we present it, we must first formally define A-submodular valuations and order neutrality.

Definition 4.5 (A-submodular)**.** Given a set of integers A, A valuation function v_i is A-submodular if: (a) $v_i(\emptyset) = 0$, (b) for any $o \in O$ and $S \subseteq O \setminus \{o\}$, $v_i(S \cup \{o\}) - v_i(S) \in A$, and (c) for any $o \in O$ and $S \subseteq T \subseteq O \setminus \{o\}$, $v_i(S \cup \{o\}) - v_i(S) \geq v_i(T \cup \{o\}) - v_i(T)$. In simple words, the valuation function is submodular, and the marginal gains are restricted to values in A.

Definition 4.6 (Order Neutrality)**.** A submodular function v_i is order neutral if for all subsets $S \subseteq O$ and two permutations of the items in S, $\pi, \pi' : [|S|] \to S$, the multi-set $\{v_i(\bigcup_{j\in[k]} \pi(j)) - v_i(\bigcup_{j\in[k-1]} \pi(j))\}_{k\in[|S|]}$ is identical to the multi-set $\{v_i(\bigcup_{j\in[k]} \pi'(j)) - v_i(\bigcup_{j\in[k-1]} \pi'(j))\}_{k\in[|S|]}$.

In simple words, the order in which the items are added to the set does not affect the marginal gains of the set of items.

We now present our results. We first show that when agents have $\{-1, 0, 1\}$-submodular valuations, computing an MEW allocation is intractable.

Theorem 4.7. *When agents have $\{-1, 0, 1\}$-submodular valuations, computing an MEW allocation is* NP*-hard.*

Proof. We reduce from the NP-complete restricted exact 3 cover problem [30].

Name: Restricted Exact 3 Cover (RX3C)
Given: A finite set of elements $U = \{1, 2, \ldots, 3k\}$, and a collection of 3-element subsets of U (denoted by $\mathcal{F}$) such that each element in U appears in exactly 3 subsets in $\mathcal{F}$.
Question: Does there exist a set of triples $\mathcal{F}' \subset \mathcal{F}$ such that every element in U occurs in exactly one subset in $\mathcal{F}'$?

Note that $|\mathcal{F}| = 3k$ since each of the $3k$ elements in U appear in exactly 3 sets in $\mathcal{F}$. Given an instance of RX3C, we construct a fair allocation instance with $3k$ agents and $9k$ items.

The $9k$ items are defined as follows: For each element $i \in U$, we have two items i and i' corresponding to the element. We also have k *cover* items and $2k$ *padding* items.

The $3k$ agents are defined as follows: For each subset $F = \{i, j, k\}$ in $\mathcal{F}$, we have an agent who has the valuation function v_F. We describe this valuation function in terms of its marginal gains to make it clear that it is submodular. If the bundle does not contain a cover item, the first item corresponding to the elements i, j, and k added to the bundle have a marginal value of 0. The second item adds a marginal value of -1. So $v_F(\{i, j, k\}) = 0$ but $v_F(\{i, i', j, k\}) = -1$. This marginal gain of 0 occurs only if the bundle does not contain a cover item; otherwise the marginal value is -1.

The cover items add a marginal value of 1 when added to an empty bundle. Otherwise they add a marginal value of 0. All padding items add a marginal value of 1, irrespective of the bundle they are added to. All other marginal values are -1.

If the original RX3C instance admits an exact cover, we can construct an allocation with egalitarian welfare 1. If the cover consists of the set of triples $\mathcal{F}' \subseteq \mathcal{F}$, we give all the agents in $\mathcal{F}'$ cover items, and all the agents outside $\mathcal{F}'$ a padding item along with a copy of each element the subset contains.

If the original RX3C instance does not admit an exact cover, then assume for contradiction that an allocation X achieves an egalitarian welfare of at least 1. Since there are only $3k$ items that provide a marginal value of 1, each agent must receive exactly one

of these items at the marginal value of 1 such that no other item in their bundle provides a marginal value of -1.

This implies the set of agents who receive a cover item must not receive any other item. Let this set of agents be $\mathcal{F}'$, and since there are k cover items we know $|\mathcal{F}'| = k$. Since $\mathcal{F}'$ is not an exact cover, there must be at least one element i present in two subsets in $\mathcal{F}'$. This implies at least one of the items corresponding to the element i must be allocated at a marginal value of -1. This is a contradiction, giving us our required separation. □

The proof of Theorem 4.7 (or at the very least, this proof technique) does not extend beyond $\{-1, 0, 1\}$-submodular valuations to $\{-1, 0, c\}$-submodular valuations. It turns out, rather surprisingly, that MEW allocations can be computed efficiently when agents have $\{-1, 0, c\}$-submodular valuations with $c \geq 2$. We show this by proving that all $\{-1, 0, c\}$-submodular valuations are order neutral, thereby showing they fall under the class of valuations for which Cousins et al. [21] present an efficient algorithm.

Proposition 4.8. *When $c \geq 2$, all $\{-1, 0, c\}$-submodular valuations are order neutral.*

Proof. Let v_i be a $\{-1, 0, c\}$ submodular valuation for $c \geq 2$. Consider some bundle, some order π over the items in the bundle, and some items o, o' which appear consecutively in the order π. That is, π consists of the set of items S (in some order) followed by the items o and o' followed by another set of items S' (in some order).

If we swap o and o', exactly two marginal values change. More specifically, $v_i(S + o) - v_i(S)$ and $v_i(S + o + o') - v_i(S + o)$ become $v_i(S + o') - v_i(S)$ and $v_i(S + o + o') - v_i(S + o')$. The value of the bundle remains the same no matter which order we use, so we must have

$$\begin{aligned}&\big(v_i(S + o') - v_i(S)\big) + \big(v_i(S + o + o') - v_i(S + o')\big)\\ &\quad = \big(v_i(S + o) - v_i(S)\big) + \big(v_i(S + o + o') - v_i(S + o)\big)\end{aligned}$$

The statement follows from noting that when $c \geq 2$, every value of $v_i(S+o+o') - v_i(S)$ has a unique decomposition into two values. More specifically, the value of $v_i(S + o + o') - v_i(S)$ can only be $-2, -1, 0, c - 1, c$ or $2c$. In each case the marginal gains of the two items are encoded by exactly the same two values. For example, the only possible way $v_i(S + o + o') - v_i(S) = c - 1$ is if one of the items provides a value of c and the other provides a value of -1; if it is -2 then both items offer a marginal gain of -1. Therefore, the set $\{v_i(S + o) - v_i(S), v_i(S + o + o') - v_i(S + o)\}$ must be exactly equivalent to the set $\{v_i(S + o') - v_i(S), v_i(S + o + o') - v_i(S + o')\}$ if they sum up to the same value. This implies that swapping two consecutive elements in an order retains order neutrality.

Since we can move from any order π to any order π' using consecutive element swaps (as is done in bubble sort), v_i must be order neutral.

This proves the statement. We note interestingly that this argument does not hold for $\{-1, 0, 1\}$ submodular valuations, since $\{-1, 1\}$ and the set $\{0, 0\}$ have the same sum. So if $\{v_i(S + o) - v_i(S), v_i(S + o + o') - v_i(S + o)\} = \{-1, 1\}$, swapping the two items could lead to the set of marginal gains $\{0, 0\}$. We use this specific property to show NP-hardness for $\{-1, 0, 1\}$ valuations in Theorem 4.7. □

5 CONCLUSIONS AND FUTURE WORK

In this work, we almost completely characterize the complexity of computing max Nash welfare and max egalitarian welfare allocations under ternary valuations. Rather unfortunately, we show that existing algorithms that work under binary and bivalued valuations cannot be generalized beyond bivalued valuations. Specifically, our results highlight a fundamental limitation of the path augmentation technique used heavily to design algorithms for binary and bivalued valuations [1, 9, 12, 13, 39, 40].

There are two natural questions left for future work. The first is the complexity of computing MEW allocations under $\{-2, 0, c\}$ valuations. Resolving this question would complete our characterization. The second question is to gain a further understanding of the increase in hardness as we generalize beyond additive and into submodular valuations. We know from Theorem 4.7 that some problems become significantly harder as we move from additive to submodular valuations but the results of Babaioff et al. [7] suggests that some problems still remain easy. There are two specific cases whose complexity still remain open questions. The first is computing MEW allocations under $\{-2, 0, c\}$ submodular valuations and the second is computing MNW allocations under $\{2, c\}$ submodular valuations. Resolving these two cases would result in a complete characterization of max Nash welfare and max egalitarian welfare allocations under submodular valuations as well.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for useful feedback. Research done while Fitzsimmons was on sabbatical visiting the University of Massachusetts, Amherst. Viswanathan and Zick are supported by an NSF grant RI-2327057. Fitzsimmons is supported in part by NSF grant CCF-2421978.

REFERENCES

[1] Hannaneh Akrami, Bhaskar Ray Chaudhury, Martin Hoefer, Kurt Mehlhorn, Marco Schmalhofer, Golnoosh Shahkarami, Giovanna Varricchio, Quentin Vermande, and Ernest van Wijland. 2022. Maximizing Nash Social Welfare in 2-Value Instances. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*. 4760–4767.

[2] Hannaneh Akrami, Bhaskar Ray Chaudhury, Martin Hoefer, Kurt Mehlhorn, Marco Schmalhofer, Golnoosh Shahkarami, Giovanna Varricchio, Quentin Vermande, and Ernest van Wijland. 2022. *Maximizing Nash Social Welfare in 2-Value Instances: The Half-Integer Case.* Tech. Rep. arXiv:2207.10949 [cs.GT]. arXiv.org. https://doi.org/10.48550/ARXIV.2207.10949

[3] Georgios Amanatidis, Haris Aziz, Georgios Birmpas, Aris Filos-Ratsikas, Bo Li, Hervé Moulin, Alexandros A. Voudouris, and Xiaowei Wu. 2023. Fair division of indivisible goods: Recent progress and open questions. *Artificial Intelligence* 322 (2023), 103965.

[4] Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, Alexandros Hollender, and Alexandros A. Voudouris. 2021. Maximum Nash welfare and other stories about EFX. *Theoretical Computer Science* 863 (April 2021), 69–85.

[5] Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. 2017. Combinatorial Algorithm for Restricted Max-Min Fair Allocation. *ACM Transactions on Algorithms* 13, 3, Article 37 (May 2017), 28 pages.

[6] Haris Aziz, Jeremy Lindsay, Angus Ritossa, and Mashbat Suzuki. 2023. Fair Allocation of Two Types of Chores. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. 143–151.

[7] Moshe Babaioff, Tomer Ezra, and Uriel Feige. 2021. Fair and Truthful Mechanisms for Dichotomous Valuations. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*. 5119–5126.

[8] Nikhil Bansal and Maxim Sviridenko. 2006. The Santa Claus Problem. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing (STOC)* (Seattle, WA, USA). 31–40.

[9] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. 2018. Greedy Algorithms for Maximizing Nash Social Welfare. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

7–13.
[10] Siddharth Barman, Sanath Kumar Krishna Murthy, and Rohit Vaish. 2018. Finding Fair and Efficient Allocations. *Proceedings of the 19th ACM Conference on Economics and Computation (EC)* (2018), 557–574.
[11] Siddharth Barman, Vishnu Narayan, and Paritosh Verma. 2023. Fair Chore Division under Binary Supermodular Costs. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. 2863–2865.
[12] Siddharth Barman and Paritosh Verma. 2021. 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)*. 169–177.
[13] Siddharth Barman and Paritosh Verma. 2022. Truthful and Fair Mechanisms for Matroid-Rank Valuations. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*. 4801–4808.
[14] Piotr Berman, Marek Karpinski, and Alexander Scott. 2003. Approximation Hardness of Short Symmetric Instances of MAX-3SAT. *Electronic Colloquium on Computational Complexity* TR03 (2003).
[15] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, and Junxing Wang. 2016. The Unreasonable Fairness of Maximum Nash Welfare. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*. 305–322.
[16] Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. 2009. On Allocating Goods to Maximize Fairness. In *Proceedings of the 50th Symposium on Foundations of Computer Science (FOCS)*. 107–116.
[17] Deeparnab Chakrabarty, Sanjeev Khanna, and Shi Li. 2015. On $(1, \epsilon)$-Restricted Assignment Makespan Minimization. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1087–1101.
[18] Miroslav Chlebík and Janka Chlebíková. 2006. Complexity of approximating bounded variants of optimization problems. *Theoretical Computer Science* 354, 3 (2006), 320–338.
[19] Richard Cole and Vasilis Gkatzelis. 2015. Approximating the Nash Social Welfare with Indivisible Items. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*. 371–380.
[20] Cyrus Cousins, Vignesh Viswanathan, and Yair Zick. 2023. Dividing Good and Great Items Among Agents with Bivalued Submodular Valuations. In *Proceedings of the 19th Conference on Web and Internet Economics (WINE)*. 225–241.
[21] Cyrus Cousins, Vignesh Viswanathan, and Yair Zick. 2023. The Good, the Bad and the Submodular: Fairly Allocating Mixed Manna Under Order-Neutral Submodular Preferences. In *Proceedings of the 19th Conference on Web and Internet Economics (WINE)*. 207–224.
[22] Shahar Dobzinski, Wenzheng Li, Aviad Rubinstein, and Jan Vondrák. 2024. A Constant-Factor Approximation for Nash Social Welfare with Subadditive Valuations. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*. 467–478.
[23] Soroush Ebadian, Dominik Peters, and Nisarg Shah. 2022. How to Fairly Allocate Easy and Difficult Chores. In *Proceedings of the 21st International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. 372–380.
[24] Zack Fitzsimmons, Vignesh Viswanathan, and Yair Zick. 2024. On the Hardness of Fair Allocation under Ternary Valuations. arXiv:2403.00943 [cs.GT] https://arxiv.org/abs/2403.00943
[25] Jugal Garg, Martin Hoefer, and Kurt Mehlhorn. 2018. Approximating the nash social welfare with budget-additive valuations. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2326–2340.
[26] Jugal Garg, Edin Husić, Wenzheng Li, László A. Végh, and Jan Vondrák. 2023. Approximating Nash Social Welfare by Matching and Local Search. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC)*. 1298–1310.
[27] Jugal Garg, Edin Husić, and László A. Végh. 2021. Approximating Nash Social Welfare under Rado Valuations. In *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing (STOC)*. 1412–1425.
[28] Jugal Garg and Aniket Murhekar. 2021. Computing Fair and Efficient Allocations with Few Utility Values. In *Proceedings of the 14th International Symposium on Algorithmic Game Theory (SAGT)*. 345–359.
[29] Jugal Garg, Aniket Murhekar, and John Qin. 2022. Fair and efficient allocations of chores under bivalued preferences. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*. 5043–5050.
[30] T. Gonzalez. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science* 38 (1985), 293–306.
[31] Daniel Halpern, Ariel D. Procaccia, Alexandros Psomas, and Nisarg Shah. 2020. Fair Division with Binary Valuations: One Rule to Rule Them All. In *Proceedings of the 16th Conference on Web and Internet Economics (WINE)*. 370–383.
[32] Pallavi Jain and Rohit Vaish. 2024. Maximizing Nash Social Welfare under Two-Sided Preferences. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI)*.
[33] Euiwoong Lee. 2017. APX-Hardness of Maximizing Nash Social Welfare with Indivisible Items. *Inform. Process. Lett.* 22 (June 2017), 17–20.
[34] J. K. Lenstra, D. B. Shmoys, and É. Tardos. 1990. Approximation Algorithms for Scheduling Unrelated Parallel Machines. *Math. Program.* 46, 3 (Feb. 1990), 259–271.
[35] Wenzheng Li and Jan Vondrak. 2021. A constant-factor approximation algorithm for Nash Social Welfare with submodular valuations. In *Proceedings of the 62nd Symposium on Foundations of Computer Science (FOCS)*. 25–36.
[36] Christos H. Papadimitriou and Mihalis Yannakakis. 1991. Optimization, approximation, and complexity classes. *J. Comput. System Sci.* 43, 3 (1991), 425–440.
[37] Erez Petrank. 1994. The hardness of approximation: Gap location. *Computational Complexity* 4 (June 1994), 133–157.
[38] Benjamin Plaut and Tim Roughgarden. 2020. Almost envy-freeness with general valuations. *SIAM Journal on Discrete Mathematics* 34, 2 (2020), 1039–1068.
[39] Vignesh Viswanathan and Yair Zick. 2023. A General Framework for Fair Allocation under Matroid Rank Valuations. In *Proceedings of the 24th ACM Conference on Economics and Computation (EC)*. 1129–1152.
[40] Vignesh Viswanathan and Yair Zick. 2023. 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)*. 179–187.