Improved MMS Approximations for Few Agent Types

Parnian Shahkar[1] and **Jugal Garg**[2]
[1]University of California, Irvine
[2]University of Illinois at Urbana-Champaign
shahkarp@uci.edu, jugal@illinois.edu

Abstract

We study fair division of indivisible goods under the *maximin share* (MMS) fairness criterion in settings where agents are grouped into a small number of *types*, with agents within each type having identical valuations. For the special case of a single type, an exact MMS allocation is always guaranteed to exist. However, for two or more distinct agent types, exact MMS allocations do not always exist, shifting the focus to establishing the existence of approximate-MMS allocations. A series of works over the last decade has resulted in the best-known approximation guarantee of $\frac{3}{4} + \frac{3}{3836}$. In this paper, we improve the approximation guarantees for settings where agents are grouped into two or three types, a scenario that arises in many practical settings. Specifically, we present novel algorithms that guarantee a $\frac{4}{5}$-MMS allocation for two agent types and a $\frac{16}{21}$-MMS allocation for three agent types. Our approach leverages the MMS partition of the majority type and adapts it to provide improved fairness guarantees for all types.

1 Introduction

Fair division of resources is a fundamental problem in various multi-agent settings. In this work, we focus on the discrete setting, where a set of indivisible goods needs to be partitioned among agents with additive preferences. The maximin share (MMS) is a widely studied fairness notion in this context. An allocation is said to satisfy MMS fairness if every agent receives a bundle of goods that they value at least their MMS value. The MMS value for an agent represents the maximum value they can guaranteee for themselves by partitioning all goods into n bundles (one for each agent) and receiving the least-valued bundle, where n is the total number of agents. Agents with identical valuation functions are considered the same *type*.

While MMS allocations provide a strong fairness guarantee for all agents, they are not guaranteed to exist for all instances [Procaccia and Wang, 2014]. Notably, when all agents have identical valuation functions—i.e., they belong to a single type—an MMS allocation is trivially guaranteed. However, in settings with three or more agents and just two distinct types, MMS allocations may not always exist [Feige *et al.*, 2021].

These non-existence results have shifted the focus toward approximate MMS guarantees. In this framework, an allocation is called α-MMS if every agent receives a bundle they value at least α times their MMS value. Over the last decade, extensive research has established the existence of allocations with an approximation guarantee of $\frac{3}{4} + \frac{3}{3836}$-MMS; see, e.g., [Barman and Krishnamurthy, 2020; Ghodsi *et al.*, 2018; Garg *et al.*, 2019; Amanatidis *et al.*, 2017a; Kurokawa *et al.*, 2018; Garg and Taki, 2021; Akrami and Garg, 2024]. This remains the best-known guarantee, even in settings with agents grouped into two distinct types.

Improving MMS approximations has proven to be a significant challenge, with progress being relatively slow. It remains unclear how to substantially improve the approximation ratio beyond $\frac{3}{4}$ for all instances. The existence of $\frac{3}{4}$-MMS was first established in [Ghodsi *et al.*, 2018], and despite extensive efforts by multiple researchers, the ratio has only been slightly improved over the past eight years. This naturally leads to the question: *Can we achieve better approximation ratios for intermediate cases where the number of agent types lies between 1 and n?*

In this paper, we address this question affirmatively by providing novel algorithms that achieve a $\frac{4}{5}$-MMS allocation for two agent types, and a $\frac{16}{21}$-MMS allocation for three agent types. Beyond their theoretical significance, these results hold practical importance, as many real-world resource allocation scenarios involve agents grouped into a few distinct categories based on shared preferences or needs. For example, co-working spaces may group users as freelancers, startups, or larger companies, while land allocation might involve agricultural, residential, or commercial priorities. These cases often involve just two or three types of agents, making it crucial to design allocation mechanisms that exploit this structure for better outcomes. These special cases have also been explored in several prior works, often motivated by additional intriguing applications; see, e.g., [Garg *et al.*, 2023; Garg *et al.*, 2024; Ghosal *et al.*, 2025].

A key innovation in our work lies in leveraging the 1-MMS partition for valuation function of the majority type, the type with the largest number of agents. We adapt this partition to achieve better approximations. This departs sig-

nificantly from traditional methods that typically begin with variations of bag-filling algorithms. Similar to [Akrami and Garg, 2024], we aim to ensure that each allocated bag contains exactly one high-valued item, enabling tighter bounds for subsequent bag-filling steps. However, rather than constructing these bags from scratch, we modify the initial MMS partition of the majority type.

Specifically, if some bags contain multiple high-valued items while others have none, we swap items to ensure that every bag contains exactly one high-valued item. The details of this process are provided in Section 3. While this process may compromise the 1-MMS property for the majority type, we ensure that the value of each modified bag remains at least $\frac{4}{5}$ times the MMS for the majority type. Therefore, any of these bags can be allocated to the agents from majority type, but the other types will determine which bags. Essentially, we prioritize assigning bags that are considered low-valued by the other types to the majority type, thereby maximizing the remaining value for the non-majority types.

When there are three types of agents, the allocation problem becomes more complicated, as the two non-majority types might disagree on which bags should be allocated to the majority type. In such cases, we categorize the bags into four classes based on their value to the non-majority types: valuable to both, valuable to one but not the other, valuable to neither. By comparing the number of bags in these categories with the number of agents in each type, we develop tailored solutions for all possible cases. The most challenging scenario arises when many bags are low-valued for both non-majority types, leaving them undesired by either. In such instances, we aggregate all items from these bags and employ a sophisticated bag-filling algorithm to ensure that all agents receive allocations that meet their MMS requirements. The algorithms that achieve improved approximation guarantees for scenarios with two or three agent types are detailed in Section 4.

While our approach is effective for instances with two and three types, it faces scalability challenges as the number of types increases. Specifically, the number of bag categories grows significantly, and the resulting case analysis becomes increasingly intricate. Therefore, this paper focuses exclusively on scenarios involving two or three agent types. Notably, barring the MMS value computation for each type through a PTAS [Woeginger, 1997], all our algorithms run in polynomial time in n and m.

In Section 2, we give formal definitions, notations, and preliminaries. Proofs for all claims, lemmas, propositions, and theorems marked with a † can be found in the supplementary material.

1.1 Additional Related Work

Given the intense study of the MMS fairness notion and its special cases and variants, we focus here on closely related work. Computing the MMS value of an agent is NP-hard. However, a Polynomial Time Approximation Scheme (PTAS) exists for this computation [Woeginger, 1997]. As noted earlier, MMS allocations are not guaranteed to exist for more than two agents with two distinct types [Procaccia and Wang, 2014; Feige *et al.*, 2021]. This non-existence has motivated the exploration of approximate MMS allocations to ensure their existence. A series of works has established the current best approximation factor of $\frac{3}{4} + \frac{3}{3836}$ for all instances [Akrami and Garg, 2024].

Several works have examined special cases. For example, when $m \leq n + 3$, where m is the total number of goods, an MMS allocation always exists [Amanatidis *et al.*, 2017b]. This bound was later improved to $m \leq n + 5$ [Feige *et al.*, 2021]. For $n = 2$, MMS allocations always exist [Bouveret and Lemaître, 2016]. For $n = 3$, the MMS approximation was improved from $\frac{3}{4}$ [Procaccia and Wang, 2014] to $\frac{7}{8}$ [Amanatidis *et al.*, 2017b] to $\frac{8}{9}$ [Gourvès and Monnot, 2019], and then to $\frac{11}{12}$ [Feige and Norkin, 2022a]. For $n = 4$, $\frac{4}{5}$-MMS allocations exist [Ghodsi *et al.*, 2018]. For $n \geq 5$, the best known factor is the general $\frac{3}{4} + \frac{3}{3836}$ bound [Akrami and Garg, 2024]. For the special case of (non-personalized) bivalued instances, MMS allocations are known to exist [Feige, 2022]. Finally, given only three agents, an $\frac{11}{12}$-MMS exists [Feige and Norkin, 2022b], and given two agents, the cut-and-choose protocol guarantees a 1-MMS allocation.

Chores The MMS notion can naturally be defined for the fair division of chores, where items provide negative value. As with goods, MMS allocations for chores do not always exist [Aziz *et al.*, 2017; Feige *et al.*, 2021]. However, substantial research on approximate MMS allocations for chores has yielded significant results. Notable works [Aziz *et al.*, 2017; Barman and Krishnamurthy, 2020; Huang and Lu, 2021; Huang and Segal-Halevi, 2023] have led to the existence of $\frac{13}{11}$-MMS allocations. For three agents, $\frac{19}{18}$-MMS allocations exist [Feige and Norkin, 2022a], and for factored instances, MMS allocations are guaranteed [Garg *et al.*, 2025]. Additionally, for the special case of personalized bivalued instances, $\frac{15}{13}$-MMS allocations exist [Garg *et al.*, 2025].

2 Preliminaries

In this paper, we primarily follow the notations used in previous work [Akrami *et al.*, 2023] to be consistent with the literature. However, beginning with Definition 6, we introduce new notations that are specific to our work.

For any positive integer n, let $[n] = \{1, 2, \ldots, n\}$, and for two positive integers i, j where $i < j$, let $[i, j] = \{i, i + 1, \cdots, j\}$. A fair division instance $\mathcal{I} = (N, M, \mathcal{V})$ consists of a set of agents $N = [n]$, a set of goods $M = [m]$, and a vector of valuation functions $\mathcal{V} = (v_1, v_2, \ldots, v_n)$. Each valuation function $v_i : 2^M \to \mathbb{R}_{\geq 0}$ represents agent i's preference over subsets of goods. We assume additive valuations, so for all $S \subseteq M$, $v_i(S) = \sum_{g \in S} v_i(\{g\})$. For ease of notation, for all $g \in M$, we use $v_i(g)$ or $v_{i,g}$ instead of $v_i(\{g\})$. Likewise, throughout this paper, we use the notation $v_i(g, g')$ as a shorthand for $v_i(\{g, g'\})$.

For a set S of goods and a positive integer d, let $\Pi_d(S)$ be the set of all partitions of S into d bundles. The maximin share (MMS) value for a valuation v is defined as:

$$\mathrm{MMS}_v^d(S) = \max_{P \in \Pi_d(S)} \min_{j=1}^{d} v(P_j).$$

When the instance $\mathcal{I} = (N, M, \mathcal{V})$ is clear from the context, we use the notation $\mathrm{MMS}_{v_i}^n(M)$ as $\mathrm{MMS}_i(\mathcal{I})$ or MMS_i.

For agent i, an MMS partition $P^i = (P^i_1, P^i_2, \ldots, P^i_n)$ satisfies $\text{MMS}_i = \min_{j\in[n]} v_i(P^i_j)$. An allocation X is MMS if $v_i(X_i) \geq \text{MMS}_i$ for all $i \in N$. More generally, for $0 < \alpha \leq 1$, an allocation X is α-MMS if $v_i(X_i) \geq \alpha \cdot \text{MMS}_i$ for all $i \in N$.

For a list of items H, let $|H|$ denote the number of items in the list. Given positive integers $i \in [|H|]$ and $j \in [|H|]$ with $i < j$, let $H[i:j]$ denote the subset of items in H from the i-th to the j-th position, inclusive. Also, let $H[-1]$ represent the last item in the list.

Definition 1 (Ordered instance)**.** *An instance $\mathcal{I} = (N, M, \mathcal{V})$ is ordered if there exists a permutation of the goods $(g_1, g_2, \ldots, g_m)$ such that for all agents $i \in N$, $v_i(g_1) \geq v_i(g_2) \geq \ldots \geq v_i(g_m)$. For any fair division instance $\mathcal{I} = ([n], [m], \mathcal{V})$, the transformation $\texttt{order}(\mathcal{I})$ produces an ordered instance $\mathcal{I}' = ([n], [m], \mathcal{V}')$, where for each $i \in [n]$ and $j \in [m]$, $v'_i(j)$ is the jth largest value in the multiset $\{v_i(g) \mid g \in [m]\}$.*

Theorem 1 (Theorem 2 in [Barman and Krishnamurthy, 2020])**.** *Given an instance $\mathcal{I}$ and an α-MMS allocation of $\texttt{order}(\mathcal{I})$, one can compute an α-MMS allocation of $\mathcal{I}$ in polynomial time.*

Theorem 1 implies the transformation `order` is *α-MMS-preserving*. For ordered instances $\mathcal{I}$, we assume without loss of generality that $v_i(1) \geq v_i(2) \geq \ldots \geq v_i(m)$ for all $i \in [n]$.

Definition 2 (Normalized instance)**.** *An instance $\mathcal{I} = (N, M, \mathcal{V})$ is* normalized *if for all agents $i \in N$ and all bundles P^i_j in an MMS partition of i, $v_i(P^i_j) = 1$. For any fair division instance $\mathcal{I}$, the transformation $\texttt{normalize}(\mathcal{I})$ computes a normalized instance $\mathcal{I}' = (N, M, \mathcal{V}')$ by determining the MMS partition P^i for each agent $i \in N$ and rescaling valuations: for all $j \in [n]$ and $g \in P^i_j$, set $v'_{i,g} = v_{i,g}/v_i(P^i_j)$.*

Lemma 1 (Lemma 4 in [Akrami *et al.*, 2023])**.** *Let $\mathcal{I}' = ([n], [m], \mathcal{V}') = \texttt{normalize}(\mathcal{I} = ([n], [m], \mathcal{V}))$. Then for any allocation A, $v_i(A_i) \geq v'_i(A_i) \cdot \text{MMS}^n_{v_i}$ for all $i \in N$.*

Note that in a normalized instance, the MMS value for every agent is 1. Consequently, Lemma 1 establishes that `normalize` is α-MMS-preserving. This means that if an allocation A is an α-MMS allocation for the normalized instance $\mathcal{I}'$, then A is also an α-MMS allocation for the original instance $\mathcal{I}$. Note that in a normalized instance, the total value of all items satisfies $v'_i([m]) = \sum_{j\in[n]} v'_i(P^i_j) = n$ for every agent $i \in [n]$. Also, for each agent i and for every MMS partition Q of agent i, we have $v'_i(Q_j) = 1 \; \forall j \in [n]$.

Given an instance $\mathcal{I}$, a reduction rule $R(\mathcal{I})$ allocates a subset $S \subseteq M$ of goods to an agent i and produces a new instance $\mathcal{I}' = (N \setminus \{i\}, M \setminus S, \mathcal{V})$.

Definition 3 (Valid reductions)**.** *A reduction rule R is a* valid α-reduction *if, for $R(\mathcal{I}) = (N', M', \mathcal{V})$, where $\{i\} = N \setminus N'$ and $S = M \setminus M'$:*

1. *$v_i(S) \geq \alpha \cdot \textit{MMS}^{|N|}_{v_i}(M)$, and*
2. *$\textit{MMS}^{|N|-1}_{v_j}(M') \geq \textit{MMS}^{|N|}_{v_j}(M)$ for all $j \in N'$.*

If R is a valid α-reduction and an α-MMS allocation A exists for $R(\mathcal{I})$, then an α-MMS allocation for $\mathcal{I}$ can be constructed by allocating S to i and distributing the remaining goods as in A. We now describe three standard transformations, known as *reduction rules*, and demonstrate their validity.

Definition 4 (Reduction rules)**.** *Consider an ordered fair division instance (N, M, v), where $M := \{g_1, \ldots, g_{|M|}\}$ and $v_{i,g_1} \geq \ldots \geq v_{i,g_{|M|}}$ for every agent i. Define*

1. *$S_1 := \{g_1\}$.*
2. *$S_2 := \{g_{|N|}, g_{|N|+1}\}$ if $|M| \geq |N| + 1$, else $S_2 := \emptyset$.*
3. *$S_3 := \{g_{2|N|-1}, g_{2|N|}, g_{2|N|+1}\}$ if $|M| \geq 2|N|+1$, else $S_3 := \emptyset$.*

*Reduction rule $R_k(\alpha)$: If $v_i(S_k) \geq \alpha \cdot \text{MMS}_i$ for some agent i, then give S_k to i. A fair division instance is called $R_k(\alpha)$-*irreducible *if $R_k(\alpha)$ cannot be applied, i.e., $v_i(S_k) < \alpha \cdot \text{MMS}_i$ for every agent i. An instance is called* totally-α-irreducible *if it is $R_k(\alpha)$-irreducible for all $k \in [3]$.*

Definition 5. *The $\texttt{reduce}_\alpha$ operation takes an ordered fair division instance as input and iteratively applies the reduction rules $R_1(\alpha)$, $R_2(\alpha)$, and $R_3(\alpha)$ in any order until the instance becomes totally-α-irreducible.*

Lemma 2 (Lemma 3.1 in [Garg and Taki, 2021])**.** *For an ordered instance and for $0 \leq \alpha \leq 1$, $R_1(\alpha)$, $R_2(\alpha)$, and $R_3(\alpha)$ are valid α-reductions.*

Lemma 3 (Lemmas 2 and 3 in [Akrami *et al.*, 2023])**.** *Let $\mathcal{I} := ([n], [m], \mathcal{V})$ be an ordered instance where $v_{i,1} \geq \ldots \geq v_{i,m}$ for each agent i. If $\mathcal{I}$ is totally-α-irreducable, then $m \geq 2n$, and for each agent i and every good $j > (k-1)n$, we have $v_{i,j} < \alpha \cdot \text{MMS}_i/k$.*

Lemma 4 (Lemma 6 in [Akrami *et al.*, 2023])**.** *Let $([n], [m], \mathcal{V})$ be an ordered and normalized fair division instance. For all $k \in [n]$ and agent $i \in [n]$, if $v_i(k) + v_i(2n - k + 1) > 1$, then $v_i(2n - k + 1) \leq 1/3$ and $v_i(k) > 2/3$.*

Definition 6. *For a fair division instance $\mathcal{I}$, define $\widehat{\mathcal{I}}_\alpha := \texttt{order}(\texttt{normalize}(\texttt{reduce}_\alpha(\texttt{order}(\mathcal{I}))))$ as the ordered, normalized, totally-α-irreducible (ONI$_\alpha$) instance of $\mathcal{I}$.*

Lemma 5. [†] *Let $\mathcal{I}$ be a fair division instance, and $\widehat{\mathcal{I}}_\alpha$ be the ONI$_\alpha$ instance of $\mathcal{I}$. $\widehat{\mathcal{I}}_\alpha$ is ordered, normalized, and totally-α-irreducible. Furthermore, the transformation of $\mathcal{I}$ to $\widehat{\mathcal{I}}_\alpha$ is α-MMS-preserving, i.e., a α-MMS allocation of $\widehat{\mathcal{I}}_\alpha$ can be used to obtain a α-MMS allocation of $\mathcal{I}$.*

Definition 7. *Let $\widehat{\mathcal{I}}_\alpha := ([n], [m], \mathcal{V})$ be an ONI$_\alpha$ instance, where $v_{i,1} \geq v_{i,2} \geq \ldots \geq v_{i,m}$ for all $i \in [n]$. The items are categorized into high-valued, middle-valued, and low-valued items as follows:*

1. *High-valued (HV) items: $HV = [n]$.*
2. *Middle-valued (MV) items: $MV = [n+1, 2n]$.*
3. *Low-valued (LV) items: If $m = 2n$[1], then $LV = \emptyset$; otherwise, $LV = [2n+1, m]$.*

[1] By Lemma 3, $m \geq 2n$.

Corollary 1 (of Lemma 3)**.** *In an ONI$_\alpha$ instance $\widehat{\mathcal{I}}_\alpha$, for each agent i and every good j: if $j \in HV$, then $v_i(j) < \alpha$; if $j \in MV$, then $v_i(j) < \alpha/2$; and if $j \in LV$, then $v_i(j) < \alpha/3$.*

Definition 8 (Single-High-Valued (SHV) Partition)**.** *Given an ONI$_\alpha$ instance $\widehat{\mathcal{I}}_\alpha := ([n], [m], \mathcal{V})$, a partition of $[m]$, $A = \{A_1, A_2, \ldots, A_n\}$ is a Single-High-Valued (SHV) partition if each bundle in A contains exactly one HV item.*

Definition 9 (SHV α-MMS Partition)**.** *Given an ONI$_\alpha$ instance $\widehat{\mathcal{I}}_\alpha := ([n], [m], \mathcal{V})$, a partition of $[m]$, $A = \{A_1, A_2, \ldots, A_n\}$ is an SHV α-MMS partition if A is an SHV partition and satisfies $v_i(A_i) \geq \alpha$ for all $i \in [n]$.*

Definition 10. *Agents are classified into types based on their valuation functions. All agents sharing the same valuation function $v : 2^m \to \mathbb{R}_{\geq 0}$ are of the same type.*

Definition 11. *A k-type instance $\mathcal{I} = ([n], [m], \mathcal{V})$ is a fair division instance involving k distinct agent types, characterized by the set of valuation functions $V = \{v_1, v_2, \ldots, v_k\}$. Each agent's valuation function belongs to this set, i.e., $\mathcal{V}_i \in V$ for all $i \in [n]$. For each $j \in [k]$, type j agents are defined as those whose valuation function is v_j.*

Remark 1. *A k-type instance can be fully described by $\mathcal{I} := ([n], [m], \{T_1, \ldots, T_k\}, \{v_1, \ldots, v_k\})$, where $\{v_1, v_2, \ldots, v_k\}$ represents the set of valuation functions for the k types, and T_j denotes the number of type j agents for any $j \in [k]$. Since each agent belongs to exactly one of the k types, $\sum_{j \in [k]} T_j = n$. We also assume that the type sizes are ordered as $T_1 \geq T_2 \geq \cdots \geq T_k$.*

Claim 1. *Let $\mathcal{I}$ be a k-type instance. The ONI$_\alpha$ instance of $\mathcal{I}$, $\widehat{\mathcal{I}}_\alpha$ is a k'-type instance where $k' \leq k$.*

Proof. In the instance $\mathcal{I}$, agents of the same type share the same valuation function. Consequently, they experience the same transformations under the operations `order`, `normalize`, and `reduce`$_\alpha$, and retain the same valuation function in $\widehat{\mathcal{I}}_\alpha$. However, agents from different types may converge to the same valuation function; for instance, if the valuations of some types are permutations of the same set of m numbers, they become identical after the `order` transformation. Consequently, the number of types can only decrease, and $\widehat{\mathcal{I}}_\alpha$ becomes a k'-type instance where $k' \leq k$. □

Definition 12. *In an α-MMS problem, given a k-type ONI$_\alpha$ instance, a type i agent is said to* claim *a bundle of items B if $v_i(B) \geq \alpha$. Furthermore, a type i is considered to claim a bundle if any type i agent claims the bundle.*

In Sections 3 and 4.1, where we study the $\frac{4}{5}$-MMS problem, type i claims a bundle B if $v_i(B) \geq \frac{4}{5}$. In Section 4.2, as we consider the $\frac{16}{21}$-MMS problem, type i claims B if $v_i(B) \geq \frac{16}{21}$.

3 SHV $\frac{4}{5}$-MMS Partition of Same-type Agents

Given a 1-type ONI$_\alpha$ instance, where v represents the common valuation function shared by all agents, the objective is to find an SHV $\frac{4}{5}$-MMS partition. Since all agents belong to the same type, the MMS value is identical for each agent, i.e., $\text{MMS}_i = \text{MMS}^n_v$ for all $i \in [n]$. Moreover, having identical valuations implies an MMS partition for any single agent serves as the MMS partition for the entire instance.

The PTAS described in [Woeginger, 1997] for computing the MMS partition of a single agent can be utilized to obtain a $(1-\epsilon)$-MMS partition in poly$(\frac{1}{\epsilon})$ time. By choosing $\epsilon = \min(0.04, \frac{1}{5n})$, we can compute a $(1-\epsilon)$-MMS partition in polynomial time. However, such a partition may include bags with multiple HV items, which violates the current problem's constraints. To address this, we design Algorithm 1 to modify the initial partition, ensuring an SHV $\frac{4}{5}$-MMS solution for same-type agents.

Main ideas of Algorithm 1: At a high level, A represents the set of *flawed* bags, while A' consists of *correct* bags. A bag B is deemed *correct* if it contains exactly one HV item and $v(B) \geq \frac{4}{5}$; otherwise, it is considered *flawed*. As long as there exists at least one flawed bag, we iteratively select a subset of flawed bags and ensure that at least one of them is corrected in each iteration.

Algorithm 1 SHV $\frac{4}{5}$-MMS of same-type agents

Input: A 1-type ONI$_\alpha$ instance $\mathcal{I} = ([n], [m], \{n\}, \{v\})$, $\alpha = \frac{4}{5}$
Output: SHV $\frac{4}{5} - \text{MMS}$.
$\epsilon \leftarrow \min(0.04, \frac{1}{5n})$.
$A^0 \leftarrow (1-\epsilon) - \text{MMS}$ partition of an agent.
$A' \leftarrow$ bags in A^0 with one HV item.
$A \leftarrow A^0 \setminus A'$.
while $|A'| < n$ **do**
    Let $a \in A$ be an arbitrary bag with $k > 1$ HV items
    Let $H = [h_1, \cdots, h_k]$ be the list of HV items in a sorted in an ascending order of valuation.
    Let $B = \{b_1, \cdots, b_{k-1}\} \subset A$ be $k-1$ bags without any HV items.
    $A \leftarrow A \setminus \{a \cup B\}$.
    Let $G = \{g_1, \cdots, g_{k-1}\}$ be $k-1$ highest valued items in $\cup_{b \in B} b$.
    Swap each HV item in $H[1 : k-1]$ with a unique item in G.
    Denote updated bags as $\bar{a}, \bar{b}_1, \cdots, \bar{b}_{k-1}$.
    if $v(\bar{a}) < \frac{4}{5}$ **then**
        $P \leftarrow$ items in $\cup_{j=1}^{k-1} \bar{b}_j$ excluding HV items $H[1 : k-1]$.
        Construct $k-1$ new empty bags $B' = \{b'_1, \cdots, b'_{k-1}\}$
        Put one HV item from $H[1 : k-1]$ in each bag of B'.
        From P fill each bag in $\bar{a} \cup B'$ until it is claimed.
        Add $\bar{a}$ and all bags of B' to A'.
    else
        Add $\bar{a}$ and any bag in $\{\bar{b}_1, \cdots, \bar{b}_{k-1}\}$ containing one HV to A', and the rest to A.
return A'

Claim 2. † *At any point in the algorithm, the following invariants hold:*

1. *The total number of bags satisfies $|A| + |A'| = n$.*

2. *For any $a \in A$, a does not contain exactly one HV item and $v(a) \geq 1 - \epsilon$.*
3. *For any $a \in A'$, a contains exactly one HV item and $v(a) \geq \frac{4}{5}$.*
4. *If there exists a bag $a \in A$ that contains k HV items, then there must be at least $k-1$ bags in A that do not contain any HV items.*

Lemma 6. † *If in some iteration of the while loop, after the swapping step, $v(\bar{a}) < \frac{4}{5}$, the algorithm never runs out of items while performing bag-filling in line 19.*

Combining these results, we arrive at the following theorem.

Theorem 2. *Given a 1-type ONI$_\alpha$ instance, Algorithm 1 returns an SHV $\frac{4}{5}$-MMS partition.*

4 Improved MMS Approximations

This section presents the main results, including Theorems 4 and 5. Together with Lemma 5 and Claim 1, these results lead to the following general theorem:

Theorem 3. *For a k-type fair division instance $\mathcal{I}$, the ONI$_\alpha$ instance of $\mathcal{I}$ results in a k'-type instance with $k' \leq k$, then:*

- *If $k' = 2$, a $\frac{4}{5}$-MMS allocation exists.*
- *If $k' = 3$, a $\frac{16}{21}$-MMS allocation exists.*

Corollary 2. *For any 2-type fair division instance, a $\frac{4}{5}$-MMS allocation exists.*

Corollary 3. *For any 3-type fair division instance, a $\frac{16}{21}$-MMS allocation exists.*

4.1 $\frac{4}{5}$-MMS for 2-type ONI$_\alpha$ Instance

Given a 2-type ONI$_\alpha$ instance, Algorithm 2 first constructs a 1-type ONI$_\alpha$ instance $\widehat{\mathcal{I}}$ based on the valuation function of the majority type. It computes the SHV $\frac{4}{5}$-MMS partition of $\widehat{\mathcal{I}}$ and sorts the n bags in ascending order according to the minority type's valuation. The first T_1 bags are assigned to type 1 agents. The remaining T_2 bags are assigned to type 2 agents if all these bags are claimed by type 2 agents. If any bag is left unclaimed, the items from all T_2 bags are pooled together. From this pool, each of the T_2 HV items is placed into a new empty bag, and the bags are filled until each of them is claimed by type 2 agents. This process is called bag-filling.

Theorem 4. *Given a 2-type ONI$_\alpha$ instance, Algorithm 2 returns a $\frac{4}{5}$-MMS.*

Proof. Note that $\alpha = \frac{4}{5}$. In Algorithm 2, first we obtain A, an SHV $\frac{4}{5}$-MMS partition of $\widehat{\mathcal{I}}$. Note that type 1 agents claim all bags in A. Then type 2 sorts all the n bags of A in an ascending order of valuation, therefore $v_2(A_j) \leq v_2(A_{j+1})$ for $j \in [n-1]$. Let $F = \{A_j\}_{j=1}^{T_1}$ be the collection of the first T_1 bags. Each bag in F is assigned to a unique type 1 agent. If $v_2(A_{T+1}) \geq \alpha$, since bags are sorted in ascending order, $v_2(A_j) \geq \alpha$ for all $j \in [T_1 + 1, n]$. Therefore, all the remained bags are claimed by type 2, and we'll assign them to the remained agents. In this case the assignment is a $\frac{4}{5}-$MMS as all the assigned bags are claimed by the corresponding agents.

Algorithm 2 $\frac{4}{5}$-MMS for 2 types

1: **Input:** 2-type ONI$_\alpha$ instance $\mathcal{I} = ([n], [m], \{T_1, T_2\}, \{v_1, v_2\}), \alpha = \frac{4}{5}$.
2: **Output:** $\frac{4}{5}-$MMS.
3: Define a 1-type ONI$_\alpha$ instance $\widehat{\mathcal{I}} = ([n], [m], \{n\}, \{v_1\})$
4: Let A be the SHV $\frac{4}{5}-$MMS partition after running Algorithm 1 on $\widehat{\mathcal{I}}$.
5: Sort bags of A by type 2 in an ascending order of valuation.
6: Assign the first T_1 bags to type 1 agents.
7: **if** type 2 agents claim all remaining bags **then**
8: Assign all the remained bags to type 2 agents.
9: **else**
10: $P \leftarrow$ items from all remained bags.
11: Put each HV item of P in a new bag.
12: From the remaining items in P, fill each new bag until a type 2 agent claims it, then assign it to her.

On the other hand if $v_2(A_{T+1}) < \alpha$, items in the remained bags are pooled in P, and since valuations are in an ascending order, $v_2(A_j) < \alpha$ for all $j \in [T_1]$. Therefore

$$\sum_{a \in F} v_2(a) < T_1 \alpha. \tag{1}$$

Note that since each bag in A has exactly one HV item, we have exactly $n - T_1$ HV items in P, and as $T_2 = n - T_1$, the number of remained HV items is exactly the same as the number of type 2 agents. We construct T_2 new empty bags, and by putting each HV item in a separate bag, the remained pool of items will not have any HV item any more. By Corollary 1 the value of each remained item in the pool is at most $\frac{\alpha}{2}$. When filling a bag until type 2 claims it, its value for type 2 cannot exceed $\frac{3\alpha}{2}$ for the following reason; before the last item was added, the bag's value was less than α, and the last item contributes at most $\frac{\alpha}{2}$. Let A_{new} be the set of bags created during the bag filling phase. We have that

$$\sum_{a \in A_{new}} v_2(a) \leq T_2 \frac{3\alpha}{2}. \tag{2}$$

Set of bags assigned to all agents is $\{A_j\}_{j=1}^{T_1} \cup A_{new}$. Putting Eq. (1) and Eq. (2) together we obtain

$$\begin{aligned} \sum_{a \in \{A_j\}_{j=1}^{T_1} \cup A_{new}} v_2(a) &< T_1 \alpha + T_2 \frac{3\alpha}{2} \\ &\leq \frac{n}{2}\alpha + \frac{n}{2} \cdot \frac{3\alpha}{2} = n \end{aligned} \tag{3}$$

where we used $\alpha = \frac{4}{5}$, $T_1 + T_2 = n$ and $T_1 \geq \frac{n}{2}$ since type 1 agents are the majority by assumption. The total value of the assigned bags is upper bounded by n, the total available value in a normalized instance. Therefore, we never run out of goods while bag filling or, equivalently, all type 2 agents receive a claimed bag. Therefore, this assignment is a $\frac{4}{5}-$MMS. □

4.2 $\frac{16}{21}-$MMS for 3-type ONI$_\alpha$ Instance

Given a 3-type ONI$_\alpha$ instance, Algorithm 3 first constructs a 1-type ONI$_\alpha$ instance $\widehat{\mathcal{I}}$ based on the valuation function of the majority type, and computes the SHV $\frac{4}{5}$-MMS partition of $\widehat{\mathcal{I}}$, A. For any choice of α, each bag in A is either liked by more than α by both types 2 and 3, or just one of them, or none. Based on the valuations of types 2 and 3, we partition the n bags in A into four classes C_1, C_2, C_3 and C_4 using Algorithm 4. By choosing $\alpha = \frac{16}{21}$, type 2 agents claim all the bags in C_2 and C_4, type 3 agents claim all the bags in C_3 and C_4, and type 1 agents claim all the bags in $A = \bigcup_{i=1}^{4} C_i$.

Finally, by comparing the number of bags in these classes with the number of agents of each type, we identify four cases. The specific approach for handling each case to achieve a $\frac{16}{21}$-MMS is detailed in Algorithm 3. Notably, the last case addressed in this algorithm presents the greatest challenge and depends on executing Algorithm 5. We analyze each case of the algorithm separately and show that, in all four cases, Algorithm 3 guarantees a $\frac{16}{21}$-MMS allocation. Consequently, we derive the following theorem.

Theorem 5. *Given a 3-type ONI$_\alpha$ instance, Algorithm 3 returns a $\frac{16}{21}$-MMS.*

Case 1: If $|C_2| > T_2$ and $|C_3| > T_3$

After allocating T_2 bags from C_2 to all type 2 agents and T_3 bags from C_3 to all type 3 agents, the remaining bags are assigned to all type 1 agents. Since all bags in A are claimed by type 1 agents, all bags in C_2 by type 2 agents, and all bags in C_3 by type 3 agents, this allocation achieves a $\frac{16}{21}$-MMS guarantee.

Case 2: If $\exists i, i' \in \{2,3\}$ s.t. $i \neq i', |C_i| > T_i$ and $|C_{i'}| \leq T_{i'}$

Let A' denote the set of T_i arbitrary bags allocated to type i agents from C_i. After this allocation, only two types of agents remain. Type 1 agents will receive T_1 bags from $A \setminus A'$. As each of the bags in A has a value of at least $\frac{4}{5}$ for type 1 agents, all of them are claimed by these agents. Since all the bags in A' are valued at less than α by type i' agents, the loss incurred by type i' is minimal. Essentially, this scenario is equivalent to considering type i' agents forfeiting these bags to type 1 agents in a two type setting where the number of type 1 agents is $T_1 + T_i$.

Lemma 7. † *If $\exists i, i' \in \{2,3\}$ s.t. $i \neq i', |C_i| > T_i$ and $|C_{i'}| \leq T_{i'}$, Algorithm 3 returns a $\frac{16}{21}$-MMS.*

Case 3: If $|C_2| \leq T_2$, and $|C_3| \leq T_3$, $|C_1| \leq T_1$

After assigning all bags in C_1, C_2, C_3 to some agents of types $1, 2, 3$ respectively, we are remained with bags in C_4 that are claimed by all types. Hence we can assign them to any remaining agents, and obtain a $\frac{16}{21}$-MMS assignment.

Case 4: If $|C_2| \leq T_2$, and $|C_3| \leq T_3$, $|C_1| > T_1$

Definition 13. *A type is considered saturated if every agent of that type has received a claimed bag in the assignment; otherwise, it is unsaturated.*

Definition 14. *A bag B is* safe *for type i agents if $v_i(B) \leq 1$.*

Algorithm 3 $\frac{16}{21}$-MMS for 3 types

1: **Input:** A 3-type ONI$_\alpha$ instance $\mathcal{I} = ([n], [m], \{T_1, T_2, T_3\}, \{v_1, v_2, v_3\})$, $\alpha = \frac{16}{21}$.
2: **Output:** $\frac{16}{21}-$MMS.
3: Define a 1-type ONI$_\alpha$ instance $\widehat{\mathcal{I}} = ([n], [m], \{n\}, \{v_1\})$
4: Let A be the SHV $\frac{4}{5}-$MMS partition after running Algorithm 1 on $\widehat{\mathcal{I}}$.
5: Using Algorithm 4, partition bags of A into C_1, C_2, C_3, C_4.
6: **if** $|C_2| > T_2, |C_3| > T_3$ **then** *// Case 1*
7: Assign T_2 bags from C_2 to all type 2 agents.
8: Assign T_3 bags from C_3 to all type 3 agents.
9: Assign the remained bags to all type 1 agents.
10: **if** $\exists i, i' \in \{2,3\}$s.t.$i \neq i', |C_i| > T_i$, $|C_{i'}| \leq T_{i'}$: **then** *// Case 2*
11: Assign T_i bags of C_i to all type i agents.
12: Type i' sorts all remained bags in an ascending order of valuation.
13: Assign the first T_1 bags to type 1 agents.
14: **if** all remained bags are claimed by type i' **then**
15: Assign all the remained bags to type i' agents.
16: **else**
17: $P \leftarrow$ items from all remained bags.
18: Put each HV item of P in a new bag.
19: Fill each new bag with remaining items from P until a type i' agent claims it, then assign it to her.
20: **else**
21: **if** $|C_1| \leq T_1$ **then** *// Case 3*
22: Assign all bags in C_1 to some type 1 agents.
23: Assign all bags in C_2 to some type 2 agents.
24: Assign all bags in C_3 to some type 3 agents.
25: Assign all remaining bags in C_4 to any remaining agents.
26: **if** $|C_1| > T_1$ **then** *// Case 4*
27: Assign T_1 bags in C_1 to all agents of type 1.
28: $P \leftarrow$ items from all remained bags.
29: Let $H = HV \cap P$, $M = MV \cap P$, $L = LV \cap P$.
30: Run Algorithm 5 with $\alpha = \frac{16}{21}$.

After assigning T_1 bags in C_1 to all agents of type 1, type 1 is saturated. Let H, M, and L denote the lists of high-valued, middle-valued, and low-valued items, respectively, that remain in the remaining $n - T_1$ bags. Each list is ordered in descending order of valuation. Since each assigned bag to type 1 had exactly one HV item, $|H| = n - T_1$. On the other hand, since any number of middle-valued items might remain after the assignment of bags to the majority type, $0 \leq |M| \leq n$. Finally, Algorithm 5 is invoked.

Main Ideas of Algorithm 5 The algorithm consists of two parts. Initially, both type 2 and type 3 are active and unsaturated. A type remains active until it meets one of the conditions specified in lines 13 or 16, or until it becomes saturated in the first part of the algorithm (by line 21). Once a type becomes inactive, it stays inactive; if none of these conditions are met, it may never become inactive.

Algorithm 4 Clustering bags of A

Input: $A = \{A_1, \cdots, A_n\}$, $\alpha = \frac{16}{21}$.
Output: A partition of bags in A into 4 classes C_1, C_2, C_3, C_4.
for $i \in \{1, 2, 3, 4\}$ **do** $C_i \leftarrow \emptyset$
for $a \in A$ **do**
    if $v_2(a) < \alpha, v_3(a) < \alpha$ **then** $C_1 \leftarrow C_1 \cup \{a\}$.
    if $v_2(a) \geq \alpha, v_3(a) < \alpha$ **then** $C_2 \leftarrow C_2 \cup \{a\}$.
    if $v_2(a) < \alpha, v_3(a) \geq \alpha$ **then** $C_3 \leftarrow C_3 \cup \{a\}$.
    if $v_2(a) \geq \alpha, v_3(a) \geq \alpha$ **then** $C_4 \leftarrow C_4 \cup \{a\}$.

As the algorithm proceeds, items are grouped into bundles that are either immediately assigned to agents or saved for future assignment. In either case, the bundled items become unavailable and are removed from H, M, and L, thereby reducing the number of available items. Consequently, at any given stage, H, M, and L represent the currently available high-valued, middle-valued, and low-valued items. The algorithm operates in two parts, as described below.

1. **First Part:** As long as there is an active type and remaining high-valued and middle-valued items, the algorithm constructs certain bundles that are *safe* for all active types. Each bundle contains one HV item and one MV item. The bags are either:
   - Assigned to an agent from an unsaturated type that claims the bag, or
   - Saved in F for the second part if no unsaturated type claims the bag.

   This process continues until no high-valued or middle-valued items remain, or no active types remain. Given a set of available high-valued and middle-valued items H and M, where $H \neq \emptyset$ and $M \neq \emptyset$, type i becomes inactive (i.e., is no longer active) if it becomes saturated (by line 21), or if $v_i(M[1], H[-1]) > 1$ and there exists no j such that $\alpha \leq v_i(M[j], H[-1]) \leq 1$, by lines 13 or 16. Recall that $H[-1]$ is the least valued available HV item.
2. **Second Part:** When both types are inactive or either $M = \emptyset$ or $H = \emptyset$, the second part begins. The collection of bags F inherited from the first part consists of bags, each containing one HV and one MV item, with a total value of less than α for any unsaturated type. If $H \neq \emptyset$, each remaining item in H is placed in a separate new bag and the bag is added to F.

   The algorithm then picks an arbitrary bag from F and fills it with remaining items until an unsaturated type claims it. The bag is then assigned to an arbitrary agent belonging to an unsaturated type that claimed it. This bag-filling procedure repeats until all types are saturated.

Proposition 1. † *Throughout the first part of the algorithm, as long as a type remains active, all saved and assigned bags are guaranteed to be safe for that type.*

Lemma 8. *In Algorithm 5, both types* 2 *and* 3 *get saturated.*

Algorithm 5 Algorithm for Case 4

Let $F \leftarrow \emptyset$, `active` $\leftarrow \{2, 3\}$, `unsaturated` $\leftarrow \{2, 3\}$. *// Part 1 begins*
while $H \neq \emptyset$ and $M \neq \emptyset$ and `active` $\neq \emptyset$ **do**
    $B \leftarrow \{M[1], H[-1]\}$.
    Let $X = \{i \in$ `active` $\mid v_i(B) > 1\}$.
    if $|X| = 0$ **then**
        if $\exists j \in$ `unsaturated`, $v_j(B) \geq \alpha$ **then**
            Assign B to a type j agent.
        else, $F \leftarrow F \cup B$.
    if $|X| = 1$ **then**
        Let i be the unique element of X.
        if $\exists j$ s.t. $\alpha \leq v_i(M[j], H[-1]) \leq 1$ **then**
            Assign $B' = \{M[j], H[-1]\}$ to a type i agent.
        else, `active` $\leftarrow$ `active` $\setminus \{i\}$.
    if $|X| = 2$ **then**
        if $\exists i \in X \mid \nexists j : \alpha \leq v_i(M[j], H[-1]) \leq 1$ **then**
            `active` $\leftarrow$ `active` $\setminus \{i\}$.
        else
            Let j be the largest index where for some $i \in X$, $\alpha \leq v_i(M[j], H[-1]) \leq 1$.
            Assign $B' = \{M[j], H[-1]\}$ to type i.
    If a bag is saved or assigned, remove its items from M and H.
    If a type is saturated, remove it from `active` and `unsaturated`. *// Part 2 begins*
Place all the remaining items of H in a separate new bag and add the bag to F.
Let $R = M \cup L$ be the pool of remained items.
while `unsaturated` $\neq \emptyset$ **do**
    Choose a bag B from F, fill it with available items from R, and assign it to any agent of type in `unsaturated` upon claim.
    Update F, R, and `unsaturated`.

By combining Lemma 8 with the fact that all bags assigned to type 1 agents have a value of at least $\frac{4}{5}$, it follows that Algorithm 3 obtains a $\frac{16}{21}$-MMS allocation in the fourth case.

5 Conclusion

In the fair division of indivisible goods, the maximin share is one of the most extensively studied fairness notions. Determining tight lower and upper bounds on the maximum α for which α-MMS allocations are guaranteed to exist remains a fundamental open problem. Since MMS allocations do not always exist even for instances with two agent types, our improved bounds represent a significant step forward in this area. To gain deeper insights into this problem, we focused on the special cases of two and three agent types. A compelling open question is whether uniform improvements can be achieved for any $k < n$ types, with guarantees that decreases as k increases, surpassing the current best-known approximation of $\frac{3}{4} + \frac{3}{3836}$.

Acknowledgements

Parnian Shahkar was supported by NSF Grant CCF 2230414. Jugal Garg has been supported by NSF Grants CCF-1942321 and CCF-2334461.

References

[Akrami and Garg, 2024] Hannaneh Akrami and Jugal Garg. Breaking the $3/4$ barrier for approximate maximin share. In *Proceedings of the the 2024 ACM-SIAM Symposium on Discrete Algorithms SODA*, pages 74–91, 2024.

[Akrami *et al.*, 2023] Hannaneh Akrami, Jugal Garg, Eklavya Sharma, and Setareh Taki. Simplification and improvement of MMS approximation. In *Proc. 32nd Intl. Joint Conf. Artif. Intell. (IJCAI)*, 2023.

[Amanatidis *et al.*, 2017a] Georgios Amanatidis, Evangelos Markakis, Afshin Nikzad, and Amin Saberi. Approximation algorithms for computing maximin share allocations. *ACM Trans. Algorithms*, 13(4), December 2017.

[Amanatidis *et al.*, 2017b] Georgios Amanatidis, Evangelos Markakis, Afshin Nikzad, and Amin Saberi. Approximation algorithms for computing maximin share allocations. *ACM Transactions on Algorithms (TALG)*, 13(4):1–28, 2017.

[Aziz *et al.*, 2017] Haris Aziz, Gerhard Rauchecker, Guido Schryen, and Toby Walsh. Algorithms for max-min share fair allocation of indivisible chores. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pages 335–341, 2017.

[Barman and Krishnamurthy, 2020] Siddharth Barman and Sanath Kumar Krishnamurthy. Approximation algorithms for maximin fair division. *ACM Transactions on Economics and Computation (TEAC)*, 8(1):1–28, 2020.

[Bouveret and Lemaître, 2016] Sylvain Bouveret and Michel Lemaître. Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Autonomous Agents and Multi-Agent Systems*, 30(2):259–290, 2016.

[Feige and Norkin, 2022a] Uriel Feige and Alexey Norkin. Improved maximin fair allocation of indivisible items to three agents. *arXiv*, abs/2205.05363, 2022.

[Feige and Norkin, 2022b] Uriel Feige and Alexey Norkin. Improved maximin fair allocation of indivisible items to three agents, 2022.

[Feige *et al.*, 2021] Uriel Feige, Ariel Sapir, and Laliv Tauber. A tight negative example for MMS fair allocations. In *Proc. 17th Conf. Web and Internet Economics (WINE)*, pages 355–372, 2021.

[Feige, 2022] Uriel Feige. Maximin fair allocations with two item values, 2022. https://www.wisdom.weizmann.ac.il/~feige/mypapers/MMSab.pdf.

[Garg and Taki, 2021] Jugal Garg and Setareh Taki. An improved approximation algorithm for maximin shares. *Artificial Intelligence*, 300:103547, 2021.

[Garg *et al.*, 2019] Jugal Garg, Peter McGlaughlin, and Setareh Taki. Approximating maximin share allocations. In *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2019.

[Garg *et al.*, 2023] Jugal Garg, Aniket Murhekar, and John Qin. New algorithms for the fair and efficient allocation of indivisible chores. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI*, pages 2710–2718, 2023.

[Garg *et al.*, 2024] Jugal Garg, Aniket Murhekar, and John Qin. Weighted EF1 and PO allocations with few types of agents or chores. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI*, pages 2799–2806, 2024.

[Garg *et al.*, 2025] Jugal Garg, Xin Huang, and Erel Segal-Halevi. Improved maximin share approximations for chores by bin packing. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2025.

[Ghodsi *et al.*, 2018] Mohammad Ghodsi, MohammadTaghi HajiAghayi, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. Fair allocation of indivisible goods: Improvements and generalizations. In *Proc. 19th Conf. Economics and Computation (EC)*, pages 539–556, 2018. (arXiv:1704.00222).

[Ghosal *et al.*, 2025] Pratik Ghosal, Vishwa Prakash HV, Prajakta Nimbhorkar, and Nithin Varma. Almost EFX for three types of agents. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, 2025.

[Gourvès and Monnot, 2019] Laurent Gourvès and Jérôme Monnot. On maximin share allocations in matroids. *Theoretical Computer Science*, 754:50–64, 2019.

[Huang and Lu, 2021] Xin Huang and Pinyan Lu. An algorithmic framework for approximating maximin share allocation of chores. In *EC '21: The 22nd ACM Conference on Economics and Computation*, pages 630–631, 2021.

[Huang and Segal-Halevi, 2023] Xin Huang and Erel Segal-Halevi. A reduction from chores allocation to job scheduling. In *Proceedings of the 24th ACM Conference on Economics and Computation EC*, page 908, 2023.

[Kurokawa *et al.*, 2018] David Kurokawa, Ariel D. Procaccia, and Junxing Wang. Fair enough: Guaranteeing approximate maximin shares. *J. ACM*, 65(2), February 2018.

[Procaccia and Wang, 2014] Ariel D Procaccia and Junxing Wang. Fair enough: Guaranteeing approximate maximin shares. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 675–692, 2014.

[Woeginger, 1997] Gerhard J Woeginger. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20(4):149–154, 1997.