12-2020

Rethinking Resource Allocation: Fairness and Computability

Andrew Searns
abs2157@rit.edu

Rethinking Resource Allocation: Fairness and Computability

by

Andrew Searns

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

Golisano College of Computing and Information Sciences
Department of Computer Science

Rochester Institute of Technology
Rochester, NY
December 2020

Rethinking Resource Allocation: Fairness and Computability

Approved by
Supervising Committee:

Dr. Hadi Hosseini, Advisor
Department of Computer Science
Rochester Institute of Technology

Dr. Ivona Bezáková, Reader
Department of Computer Science
Rochester Institute of Technology

Dr. Stanisław Radziszowski, Observer
Department of Computer Science
Rochester Institute of Technology

Contents

Abstract

In the field of multiagent systems, one important problem is fairly allocating items among a set of agents. The gold standard fairness property is envy-freeness whereby each agent prefers the bundle allocated to them over any other bundle. For indivisible goods, envy-freeness cannot be guaranteed: consider two agents and one item. In the context of indivisible goods, one key fairness notion which has gained significant attention in recent years is the maximin share guarantee (MMS). MMS extends the cut-and-choose protocol to multiple agents by guaranteeing each agent as much value as if they divided items into bundles but chose last. However, MMS is not guaranteed to exist, and even it exists, computing an MMS allocation is computationally hard. Consequently, several approximation techniques were proposed to ensure all agents receive a fraction of their MMS.

We propose an orthogonal approximation which aims to guarantee MMS for a fraction of the agents. We construct instances where any optimal approximation algorithm fails to guarantee MMS for most agents. We show how to interpolate between the fraction of agents and the fraction of MMS guaranteed to agents. We prove the existence of allocations which satisfy $\frac{2}{3}$ of the agents. Our algorithmic technique immediately implies a polynomial-time algorithm for any number of items when there are less than nine agents. Our results significantly reduce the existence gap of MMS when agents divide the items into $\frac{3n}{2}$ bundles. We empirically demonstrate that our algorithm outperforms its worst-case bounds in practice on both synthetic and real-world data.

We extend our discussion of maximin share approximations to chores. We extend many of the techniques used for MMS approximations to the chores setting. Using these new techniques, we prove the existence of allocations which satisfy all agents when they expect to divide the workload among $\frac{2n}{3}$ agents.

1 Introduction

Multiagent systems are ubiquitous in our day-to-day lives. From lines at the grocery store to voting for representatives, nearly every action we take can be seen as the result of multiple people interacting. When humans are not present, the interactions of computers and robots can be seen as a multiagent system. Even within a single computer, the allocation of registers can be viewed as a multiagent system. Due to their ubiquity, the study of multiagent systems has gained significant attention in recent years. Within the field of multiagent sytems, one key property drives a significant fraction of research: fairness. Across the subfields of voting, auction design, and item allocation, fairness is defined slightly differently, but the key idea is the same; each agent wants to be treated the same as other agents. In voting, this might mean that each agent's vote counts equally, but it might also mean that agents who vote for a non-favorable candidate still receive some influence after their candidate is removed from the election.

In the field of fair division, fairness is often easier to define. Each agent wants to receive a bundle that they prefer to the bundle of others. This notion, known as envy-freeness, is the gold standard when allocating items among agents. For divisible goods, an envy-free allocation always exists [Edward Su, 1999]. It remains an open question whether envy-free allocations can be found with only polynomially many queries for agents' valuations. In this setting, proportionality, a relaxation of envy-freeness, can be guaranteed in a linear number of queries. In a proportional allocation, each agent receives $\frac{1}{n}$th of their total valuation for the items. When items are indivisible, the problem becomes much harder. In the indivisible setting, neither envy-free nor proportional allocations are guaranteed to exist. Consider a single good with two interested agents. Since the good is indivisible, one of the agents will remain empty-handed.

In the context of indivisible items, multiple alternative notions of fairness have been proposed. One criterion which has gained significant attention in recent years is the maximin share guarantee (MMS). In an MMS allocation, each agent requires at least as much value as if they divided items into bundles and then received the least valuable bundle of the division. Intuitively, each agent tries to divide items as evenly as possible. When valuations are heavily skewed in favor of a few items, the MMS guarantee focuses more on how the remaining items can be divided fairly. Despite the apparent benefits of MMS over proportionality, it turns out that an MMS allocation does not always exist [Procaccia and Wang, 2014]. Even when it exists, finding an MMS allocation is NP-Hard even for two agents with identical preferences [Bouveret and Lemaître, 2016]. Consequently, recent research has focused on bounding the ratio of MMS that each agent can be guaranteed simultaneously [Amanatidis et al., 2017, Aziz et al., 2017, Garg and Taki, 2020, Garg et al., 2018, Ghodsi et al., 2018, Huang and Lu, 2019, Kurokawa et al., 2016]. In this research, we propose an alternative notion of fairness based one the population of agents. The overall thesis of this project is:

> *In the allocation of indivisible resources (goods or chores) among multiple agents, fairness can be guaranteed for a large subset of agents, and allocations can be computed efficiently when considering the majority of the participating agents.*

Current techniques aim to provide all agents with a constant fraction of their MMS guarantee. This approach may leave most agents without the resources they require to complete tasks. Thus instead of asking what ratio of MMS each agent can be guaranteed, we ask what ratio of agents can be guaranteed their MMS guarantee. While this may initially appear less fair, there are multiple real world scenarios where such a guarantee is applicable. For example, doctors and nurses in a hospital require a complete set of resources before any operation can be done. Thus it is more important that all necessary resources are available for a subset of operations. The alternative where a subset of resources is available for each operation may

mean that few to none of the operations can be completed successfully, which may jeopardize patient health. Similarly, course selection in colleges also necessitates this resource allocation constraint; senior level students must be guaranteed seats in major specific classes in order to prevent graduation delays. Lower level students may then be presented alternatives satisfying general education requirements if there is insufficient space in their in-major classes.

The majority of fair division research focuses on the case that items are non-negatively valued by all agents. In this context, the items are referred to as goods. However, when items are negatively valued, referred to as chores or bads, many results do not immediately extend into this domain. We explore both traditional and population-based approximations of MMS in both the goods and chores settings.

1.1 Contributions

The main contributions of this work are as follows:

- We propose a novel (α, β)-MMS fairness that considers the population of agents. Here we show that optimizing classical approximations might perform arbitrarily poor with respect to our new fairness notion.[1] We prove the existence of $(\frac{2}{3}, 1)$-MMS allocations, and our proof is algorithmic for $n < 9$. Empirically, we demonstrate that our algorithm extends beyond its theoretical bounds and computes $(\frac{2}{3}, 1)$-MMS allocations on a mixture of synthetic and real-world data where $n \geq 9$.
- Using the above result, we improve the existence gap for MMS^k from $k \leq 2n - 2$ to $k \leq \lceil \frac{3n}{2} \rceil$. This tightens the open upper-bound towards the current lower-bound of $k \geq n + 1$.
- We extend the existing framework for maximin share approximations in the chores settings. In contrast to goods, we show that a $\frac{2}{3}$ approximation based on the population of agents can easily be computed in polynomial time.

The detailed proofs and missing discussions can be found in the appendix.

[1] This result has already appeared in AAAI 2020 [Searns and Hosseini, 2020].

2 Preliminaries

A fair division **instance** $\mathcal{I} = \langle N, M, V \rangle$ consists of a set of agents $N = \{1, 2, \ldots, n\}$[2], a set of m indivisible items M, and a valuation profile V. Each agent is endowed with a valuation function $v_i : 2^M \to \mathbb{R}$. We assume that valuations are additive: for any set of items $S \subseteq M$, $v_i(S) = \sum_{g \in S} v_i(\{g\})$. For simplicity, we write $v_i(g)$ in place of $v_i(\{g\})$.

We say that an instance is a **goods instance** if each agent has non-negative value for all items. Conversely, we say that an instance is a **chores instance** if each agent has negative value for all items. Any instance which is neither a goods instance nor a chores instance is a **mixed instance**.

An **allocation** $A = (A_1, \ldots, A_n)$ is an n-partition of M (where $A_i \cap A_j = \emptyset$ for all $i, j \in N$ and $\cup_{i \in N} A_i = M$) where each agent i receives the corresponding bundle A_i. A partial allocation is an allocation of a subset of $M' \subseteq M$ of the goods. Let $\Pi_k(M)$ denote the set of k-partitions of M.

2.1 Fairness Notions

The most desirable fairness notion is that of **envy-freeness** (proposed by Foley [1967]) whereby each agent weakly prefers the bundle allocated to them over any other bundle (see Figure 1). However, this notion cannot be guaranteed for indivisible goods. The simplest relaxation of envy-freeness adds an "up to one" condition that allows agents to overlook a single item's worth of value. Formally, the envy-free up to one good (EF1) notion is defined as follows.

Definition 1 (Budish [2011]). *An allocation $A = (A_1, \ldots, A_n) \in \Pi_n(M)$ is **envy-free up to one good** (EF1) if for each pair of agents $i, j \in N$, at least one of the following holds:*

- $v_i(A_i) \geq v_i(A_j)$
- $\exists g \in A_j$ *such that* $v_i(A_i) \geq v_i(A_j \setminus \{g\})$
- $\exists b \in A_i$ *such that* $v_i(A_i \setminus \{b\}) \geq v_i(A_j)$.

While EF1 is often desirable, it sometimes allows unnecessarily skewed valuations. For example, consider two agents with three markers and a car on the market. The allocation which gives a car and a marker to one agent and the remaining two markers to the other agent is EF1 (removing the car eliminates the envy). However, a fairer solution would be to separate the car from all three markers. One notion which addresses this issue is envy-freeness up to any good (EFX). When agents compare bundles in an EFX bundle, they must be envy-free up to the least valuable good in the other agent's bundle. For example, this means that the agent with two markers would not be envy-free after removing the marker from the other agent's bundle.

Definition 2 (Caragiannis et al. [2016]). *An allocation $A = (A_1, \ldots, A_n) \in \Pi_n(M)$ is **envy-free up to any good** (EFX) if for each pair of agents $i, j \in N$, at least one of the following holds:*

- $v_i(A_i) \geq v_i(A_j)$
- $\forall g \in A_j$ *such that* $v_i(g) > 0$, $v_i(A_i) \geq v_i(A_j \setminus \{g\})$
- $\forall b \in A_i$ *such that* $v_i(b) < 0$, $v_i(A_i \setminus \{b\}) \geq v_i(A_j)$.

While EFX is very close to envy-freeness, it is only known to exist for three agents [Chaudhury et al., 2020].

Another relaxation of envy-freeness is known as proportionality (See Figure 2). Perhaps the most simple fairness notion, proportionality requires that each agent receives at least a proportional amount of their value for all items.

[2]For simplicity, we sometimes write $[k]$ to denote the set of integers $[k] = \{1, 2, \ldots, k\}$.

Agent	g_1	g_2	g_3	g_4	g_5
1	(6)	(2)	5	10	1
2	1	4	(12)	3	6
3	7	3	8	(9)	(2)

Figure 1: The circled allocation is EF1 but not EFX as good g_4 can be removed from agent 3's bundle to eliminate agent 1's envy, but not good g_5. as each agent prefers the bundle allocated to them over each other allocated bundle.

Definition 3 (Steinhaus [1949]). *An allocation $A = (A_1, \ldots, A_n) \in \Pi_n(M)$ is **proportional** (Prop) if for each agent $i \in N$, $v_i(A_i) \geq \frac{v_i(M)}{n}$.*

Agent	g_1	g_2	g_3	g_4	g_5	Total	Prop_i
1	(3)	5	(9)	1	6	24	8
2	2	6	1	(8)	7	24	8
3	10	(8)	5	3	(1)	27	9

Figure 2: The circled allocation is Prop as each agent receives at least $\frac{v_i(M)}{n}$.

Unfortunately, like envy-freeness proportionality cannot be guaranteed when items are indivisible: consider two agents and a single desirable item. The maximin share notion was introduced to help deal with this issue in proportionality (See Figure 3). When item valuations are heavily skewed, the maximin share guarantee lets agents request as close to a proportional allocation as possible.

Definition 4 (Budish [2011]). *The k **maximin share guarantee** of agent $i \in N$ is the maximum amount of value that agent i can guarantee for themself by dividing items into k bundles and then selecting their least valuable bundle. Formally,*

$$MMS_i^k(M, v_i) = \max_{(A_1,\ldots A_k)\in\Pi_k(M)} \min_{j\in[k]} v_i(A_j).$$

When M, v_i, or k are clear from context, we may omit them and write MMS_i to mean $MMS_i^k(M, v_i)$.

Agent	g_1	g_2	g_3	g_4	g_5
1	(3)	(5)	9	1	6
1	3	5	(9)	1	6
1	3	5	9	(1)	(6)

(a) The MMS^3 partition for agent 1: $\text{MMS}_1^3 = 7$.

Agent	g_1	g_2	g_3	g_4	g_5	MMS_i
1	3	(5)	9	1	(6)	7
2	2	6	(1)	(8)	7	8
3	(10)	8	5	3	1	8

(b) An MMS allocation.

Figure 3: The MMS guarantee requires two parts: computing each agent's MMS values and finding an allocation which guarantees each agent that value.

While MMS exists with high probability in random instances [Dickerson et al., 2014], it is both not guaranteed to exist [Procaccia and Wang, 2014] and NP-hard to compute even when it exists [Bouveret and Lemaître, 2016]. Consequently, recent studies in maximin share fairness aim to approximate the MMS guarantee. The most common relaxation is to ensure that each agent receives at least a β fraction of their maximin share. This idea was pushed to its limits by Heinen et al. [2018] with the notion of optimal-MMS. Given an instance, an optimal-MMS allocation is any allocation which maximizes the minimum ratio any

agent receives of their MMS (See Figure 4). Since the number of allocations is finite, an optimal-MMS allocation is guaranteed to exist.

Definition 5 (Heinen et al. [2018]). *Given an instance, the **optimal-MMS** ratio is the best possible approximation of MMS that can be achieved by all agents simultaneously. For a goods instance, the optimal-MMS ratio is defined as*

$$\lambda^* = \max_{(A_1,\dots A_n)\in\Pi_n(M)} \min_{i\in[n]} \frac{v_i(A_i)}{MMS_i^n}.$$

Likewise, for a chores instance, the optimal-MMS ratio is the minimal ratio that can be satisfied simultaneously by all agents. I.e.

$$\lambda^* = \min_{(A_1,\dots,A_n)\in\Pi_n(M)} \max_{i\in[n]} \frac{v_i(A_i)}{MMS_i^n}.$$

An allocation $A = (A_1, \dots, A_n)$ satisfies optimal-MMS if each agent receives at least a λ^* fraction of their MMS guarantee: for all $i \in N$, $v_i(A_i) \geq \lambda^* MMS_i^n$.

Agent	g_1	g_2	g_3	g_4	g_5	MMS_i
1	3	5	(9)	1	6	7
2	2	6	1	(8)	(7)	8
3	(10)	(8)	5	3	1	8

Figure 4: The circled allocation maximizes the minimum ratio $\frac{v_i(A_i)}{\text{MMS}_i}$ and thus satisfies optimal-MMS.

2.2 Population Based Fairness

In this research, we propose an orthogonal direction of maximin share approximations. We wish to approximate fairness by maximizing the number of satisfied agents. We present a notion of (α, β)-MMS wherein an α fraction of agents require a β ratio of their MMS (See Figure 5).

Definition 6. *An allocation $A = (A_1, A_2, \dots, A_n) \in \Pi_n(M)$ satisfies (α, β)-**MMS** if there exists a subset $N' \subseteq N$ such that $|N'| \geq \lfloor \alpha |N| \rfloor$ and for all $i \in N'$, $v_i(A_i) \geq \beta MMS_i^n$. We say that (α, β)-MMS exists if for every $N' \subseteq N$ with $|N'| \leq \lfloor \alpha |N| \rfloor$, there exists an allocation $A = (A_1, A_2, \dots A_n) \in \Pi_n(M)$ such that for all $i \in N'$, $v_i(A_i) \geq \beta MMS_i^n$.*

We note that the definition of (α, β)-MMS existing is equivalent if the inequality is replaced with equality. Any subset N' with size $|N'| < \lfloor \alpha |N| \rfloor$ is a subset of another set $N'' \supset N'$ such that $N'' \subseteq N$ with $|N''| = \lfloor \alpha |N| \rfloor$. Any allocation that gives each agent in N'' a β fraction of their MMS must also give each agent in N' a β fraction of their MMS. Thus that allocation may be reused to satisfy the subset N'.

	g_1	g_2	g_3	g_4	g_5	MMS_i
1	3	(5)	9	(1)	6	7
2	(2)	6	(1)	8	7	8
3	10	8	5	3	(1)	8

Figure 5: The circled allocation simultaneously satisfies $(\frac{1}{3}, \frac{6}{7})$-MMS, $(\frac{2}{3}, \frac{3}{8})$-MMS, and $(1, \frac{1}{8})$-MMS.

2.3 Efficiency Notions

While fairness is the primary objective in this research, an important tangential concern is that of economic efficiency. Fairness notions such as envy-freeness can be trivially satisfied by giving all agents nothing. However, this trivial allocation wastes significant value that could otherwise be given to agents. Economic efficiency describes the notion that there is no unnecessarily wasted value. The most common efficiency measure is that of Pareto optimality (PO).

Definition 7. *Given two allocations $A = (A_1, A_2, \ldots, A_n)$ and $B = (B_1, B_2, \ldots, B_n)$ $A, B \in \Pi_n(M)$, we say that A **Pareto dominates** B if for all $i \in N$, $v_i(A_i) \geq v_i(B_i)$ with at least one of these inequalities strict. An allocation is **Pareto optimal** if there is no allocation which Pareto dominates it.*

When we consider threshold based fairness measures (like proportionality or maximin share), we often ignore economic efficiency because empty allocations do not satisfy non-trivial thresholds. However, if an allocation exists which satisfies the threshold fairness criteria, then by repeatedly following Pareto improvements (the set of allocations is finite, so this process will terminate), we will eventually find an allocation which satisfies the threshold fairness and which is also Pareto optimal. However, finding Pareto improvements is an NP-Hard problem when valuations are additive [Aziz et al., 2019], and thus finding allocations which simultaneously satisfy fairness and efficiency is an important extension of any fairness research.

Agent	g_1	g_2	g_3	g_4	g_5
1	3	5	(9)	[1]	[6]
2	[2]	[6]	1	(8)	(7)
3	(10)	(8)	[5]	3	1

Figure 6: The squared allocation is Pareto dominated by the circled allocation. The circled allocation, however is not Pareto dominated by any allocation and is thus Pareto optimal.

3 Related Works

Problem	Existence	Computation
$n = 2$	✓ (by definition)	Cut and choose (Bouveret and Lemaître [2016])
$n = 3$	✗ Kurokawa et al. [2018]	7/8 approximation Amanatidis et al. [2017]
$n \geq 4$	✗ Kurokawa et al. [2018]	3/4 approximation Ghodsi et al. [2018]
Identical[3]	✓ (by definition)	NP-Hard (Bouveret and Lemaître [2016])
Binary	✓ Bouveret and Lemaître [2016]	Round Robin (Bouveret and Lemaître [2016])
$\{0, 1, 2\}$	✓ Amanatidis et al. [2017]	polynomial time (Amanatidis et al. [2017])

Table 1: A summary of MMS results for Goods

Results for Goods. The maximin share guarantee was first proposed by Budish [2011]. When it was first proposed, it was conjectured that an MMS allocation always exists. This idea was supported by Dickerson et al. [2014] who proved that MMS exists with high probability whenever the number of goods exceeded the number of agents by at least a logarithmic factor. This conjecture was falsified by Procaccia and Wang [2014] with a construction which used exponentially many goods. The MMS counter-example was later generalized to n agents and reduced to only $3n + 4$ goods [Kurokawa et al., 2016, 2018].

The complexity of deciding whether or not MMS exists for a given instance was studied by Bouveret and Lemaître [2016]. They gave a simple algorithm to show that MMS existence can be decided in Σ_2^P; however, this is not tight to their NP-hard lower bound. It remains an open question whether MMS existence can be decided in NP. Determining whether an MMS allocation exists appears to be based around two related questions: determining MMS values and checking whether a given allocation satisfies MMS. The former question is NP-complete but it admits a PTAS[4] to approximate MMS values [Woeginger, 1997], and the latter is known to be in coNP where a negative certificate demonstrates an MMS partition for some agent who did not receive their MMS.

In the special case that goods are valued either $\{0, 1\}$ the round robin algorithm computes an MMS allocation in polynomial time [Bouveret and Lemaître, 2016]. This result was extended to valuations from $\{0, 1, 2\}$ using a more intricate polynomial time algorithm [Amanatidis et al., 2017]. If there are only two agents, an MMS allocation exists using the cut-and-choose protocol [Bouveret and Lemaître, 2016]. Likewise, when all agents have identical preferences, any MMS partition is also an MMS allocation by definition.[5] Table 1 summarizes the existing MMS results for goods instances.

For more general instances, each agent can be guaranteed a bundle worth at least $\frac{2}{3}$ of their MMS using relatively simple algorithms [Amanatidis et al., 2017, Garg et al., 2018, Kurokawa et al., 2016]. With more intricate algorithms, this result was extended to a $\frac{3}{4} - \epsilon$ approximation by Ghodsi et al. [2018] and

[4]See Appendix A for a discussion on PTAS algorithms.

[5]This result extends to the case where all but one agents have identical preferences. Any agent with identical preferences divides the items into bundles according to their MMS partition, then the one differing agent selects a bundle first. Since at least one bundle has value at least the average, this agent will be able to select a bundle worth at least their MMS value.

a $\frac{3}{4}$ approximation by Garg and Taki [2020]. And in the case of three agents, this approximation can be extended to $\frac{7}{8}$ [Amanatidis et al., 2017]. Another recent direction of research is to focus on the maximum approximation of MMS possible given an instance. Given an instance, the optimal-MMS notion guarantees each agent the maximum possible ratio of their MMS which can be satisfied simultaneously. This notion of optimal-MMS always exists because the number of allocations is finite; the approximation ratio is guaranteed to be at least 1 if an MMS allocation exists. Heinen et al. [2018] presented a PTAS algorithm for computing an approximately optimal-MMS allocation.

The closest idea to approximating MMS with respect to the population is the l-out-of-d MMS notion presented by Babaioff et al. [2019]. This notion guarantees each agent at least as much value as if they divided items into d bundles and then selected the last l of them. Under this notion, it is known that a 1-out-of-$(2n-2)$ MMS allocation always exists Aigner-Horev and Segal-Halevi [2019]. In the l-out-of-d setting, Segal-Halevi [2019] demonstrated a dominance relationship and characterized sufficient conditions for the l-out-of-d maximin guarantee to be larger than an l'-out-of-d' maximin guarantee.

Our setting is only concerned with agent's valuations for the bundles allocated to them. A tangential setting views this allocation as a market equilibrium. In that setting, agents are given equal budgets and any allocation also requires a set of prices for which all agents can afford to buy the bundles allocated to them. The market clearing equilibrium is one where every item is bought and no agent has excess budget. In this setting, Budish [2011] proved that 1-out-of-$(n+1)$ MMS can be guaranteed with approximate market clearing. However, in our setting, excess demand in the approximate market clearing translates to over-allocation of some goods. This might mean that some goods are allocated to multiple agents despite there only being one copy of it. Thus the 1-out-of-$n+1$ allocation may not be feasible in our setting.

Results for Chores. Aziz et al. [2017] extended many of the initial results for goods to the context of chores. They proved that MMS does not exist in general in the chores setting by adapting the MMS counterexample for three agents. They also extended the NP-hardness result for computing an MMS allocation to the setting of chores. Using a simple greedy algorithm, they provided a 2-approximation of both MMS and the analogous definition of optimal-MMS.[6] Furthermore, when the number of agents is fixed, they provide a $(1+\epsilon)$ PTAS for optimal-MMS. This paper also proved a relationship between the MMS guarantee for chores with the minimax share guarantee for goods.

In polynomial time, it is possible to give each agent a $\frac{4}{3}$-approximation of MMS using algorithms similar to the $\frac{2}{3}$-approximation for goods [Amanatidis et al., 2017]. Huang and Lu [2019] proved that an $\frac{11}{9}$-approximation always exists, and using the PTAS for MMS values from Woeginger [1997], they provide a polynomial time algorithm to compute an $(\frac{11}{9}+\epsilon)\text{MMS}^n$ allocation. Table 2 shows the strongest known bounds for various approximations of MMS.

[6] In the context of chores, MMS approximations are flipped. It is better to receive a smaller approximation (closer to 1).

Problem	Goods	Chores
MMS	✗ Procaccia and Wang [2014]	✗ Aziz et al. [2017]
$(1, \beta)$-MMS	$\beta = \frac{3}{4}$ Garg and Taki [2020]	$\beta = \frac{11}{9}$ Huang and Lu [2019]
MMS$^{\gamma n}$	$\gamma = \frac{2n-2}{n}$ Aigner-Horev and Segal-Halevi [2019] $\gamma = \frac{3}{2}$ Corollary 2	$\gamma = \frac{2}{3}$ Theorem 6
$(\alpha, 1)$-MMS	$\alpha = \frac{2}{3}$ Theorem 3	$\alpha = \frac{n-1}{n}$ Observation 1

Table 2: Approximation Results for Goods and Chores

4 Maximin Share Guarantee for Goods

In this section we discuss various approximations of the maximin share guarantee for goods instances. In Section 4.1, we first prove common techniques for computing MMS approximations. We then explore the trade-offs inherent to MMS approximations in Section 4.2. In Section 4.3, we explore (α, β)-MMS for small values of n. When n is large, we provide a simple algorithm for $(\frac{1}{2}, 1)$-MMS in Section 4.4. With significantly more intricate analysis, we prove that $(\frac{2}{3}, 1)$-MMS exists in Section 4.5. In Section 4.6, we consider the relationships between different MMS approximations. Our proof of $(\frac{2}{3}, 1)$-MMS existence is algorithmic for $n < 9$, but we empirically demonstrate that it effectively extends to more agents in Section 4.7.

4.1 Warmup: Standard Lemmas

Prop Bound. In the study of maximin share approximations, most algorithms leverage a few simple lemmas. The first such lemma relates the maximin share to the proportional fair share.

Lemma 1. *For each agent $i \in N$, $MMS_i^k(M) \leq \frac{v_i(M)}{k}$.*

Proof. Suppose this is not the case. Consider an MMS partition $(M_1, \ldots, M_k) \in \Pi_k(M)$ for agent i. Since each bundle in an MMS partition is worth at least the MMS value, there are k bundles with value at least $\text{MMS}_i^k(M) > \frac{v_i(M)}{k}$. Adding these bundles implies that $v_i(M) = k\text{MMS}_i^k(M) > k\frac{v_i(M)}{k} = v_i(M)$. □

Scale Invariance. The next lemma shows that we can rescale agents' valuations as MMS allocations are scale invariant. This lemma was originally presented by Ghodsi et al. [2018].

Lemma 2 (Ghodsi et al. [2018])**.** *Let $I = \langle N, M, V \rangle$ be an instance and $c > 0$ be a real scalar. Let $I' = \langle N, M, V' \rangle$ be constructed so that $v_i'(g) = cv_i(g)$ for all $g \in M$. Then $MMS_i'^k(M) = cMMS_i^k(M)$.*

Proof. If goods are scaled so that $v_i'(g) = cv_i(g)$, then by the additivity of valuations, $v_i'(S) = \sum_{g \in S} v_i'(S) = \sum_{g \in S} cv_i(g) = c\sum g \in Sv_i(g) = cv_i(S)$. Since this is true for every bundle $S \subseteq M$, it is also true of each bundle of any partition. Thus all bundles, and therefore the lowest value bundle of a partition, are scaled simultaneously. Since this is true for every partition, we have that $\text{MMS}_i^k(M, v_i') = \max_{(A_1,\ldots,A_k)\in\Pi_k(M)} \min_{j\in[k]} v_i'(A_j) = c\max_{(A_1,\ldots,A_k)\in\Pi_k(M)} \min_{j\in[k]} v_i(A_j) = c\text{MMS}_i^k(M, v_i)$. Furthermore, if an allocation satisfies MMS I, then $v_i(A_i) \geq \text{MMS}_i^k(M, v_i)$ so $v_i'(A_i) \geq \text{MMS}_i^k(M, v_i')$ and it also satisfies MMS on I'. □

We define the function `Scale`$(I,\ k)$ which takes an input instance $I = \langle N, M, V \rangle$ and a positive real value k and outputs a new instance $I' = \langle N, M, V \rangle$ where for all agents $i \in N$, $v_i'(M) = k$. This is easily

Algorithm 4.1: Ordering an Instance

Input : Instance $I = \langle N, M, V \rangle$ on n agents
Output: An Ordered instance $I' = \langle N, M, V' \rangle$

for $j = 1$ **to** m **do**
  for $i = 1$ **to** n **do**
    Let $g =$ agent i's jth most valuable item;
    $v'_i(g_j) = v_i(g)$;

Algorithm 4.2: Unordering an Allocation

Input : Allocation $A' = (A'_1, \ldots, A'_n)$ for an ordered instance I' of I
Output: Allocation $A = (A_1, \ldots, A_n)$ for the unordered instance I

$A = (\emptyset, \ldots, \emptyset)$ and $M' = M$;
for $j = 1$ **to** m **do**
  Let agent i be the agent who received g_j in A'_i;
  Let $g = \arg\max_{g \in M'} v_i(g)$;
  $A_i = A_i \cup \{g\}$ and $M' = M' \setminus \{g\}$;

Figure 7: Algorithms for ordering an instance and unordering an allocation.

accomplished by multiplying each good by the value $\frac{k}{v_i(M)}$ because $v'_i(M) = \sum_{g \in M} v'_i(g) = \sum_{g \in M} v_i(g) \cdot \frac{k}{v_i(M)} = \frac{k}{v_i(M)} \sum_{g \in M} v_i(g) = \frac{k}{v_i(M)} v_i(M) = k$.

Ordering. Bouveret and Lemaître [2016] proved that the most difficult instances for fair division are those where agents agree on the ordinal preferences over the items.

Definition 8. *An instance $I = \langle N, M, V \rangle$ is **ordered** if there exists an ordering $(g_1, \ldots, g_m)$ such that for all $i \in N$, $v_i(g_1) \geq v_i(g_2) \geq \ldots \geq v_i(g_m)$.*

Using Algorithm 4.1, Bouveret and Lemaître [2016] showed that any instance can be converted into a related ordered instance in polynomial time. Furthermore, using Algorithm 4.2, any allocation for the ordered instance can be converted into an allocation for the original instance where each agent receives at least as much value.

Lemma 3 (Bouveret and Lemaître [2016])**.** *Let $I' = \langle N, M, V' \rangle$ be an ordered instance constructed from the original instance $I = \langle N, M, V \rangle$. Given allocation A' on I', a corresponding allocation A on I can be computed in polynomial time such that for all $i \in N, v_i(A_i) \geq v'_i(A'_i)$.*

Proof. Algorithm 4.2 allocates a single item at each iteration of the for loop. On iteration k, there have been $k-1$ item allocated. Thus among the top k item of agent i, at least one such item still remains. Thus agent i will take an item worth at least $v_i(g_k)$. Each item allocated in the unordered allocation is worth at least as much as the corresponding item allocated in the ordered allocation. By additivity, the resulting bundles in the unordered allocation are worth at least as much as the bundles of the ordered allocation. □

Reductions. The next important lemma (originally presented by [Amanatidis et al., 2017]) shows that removing a single good and any agent results in a new instance where the remaining agents MMS values are at least as large. Thus if any good has value at least $\frac{v_i(M)}{n}$ to any agent, we may give that good to the interested agent and approximate MMS in the reduced instance.

Lemma 4 (Amanatidis et al. [2017])**.** *For all $i \in N$ and $g \in M$, $MMS_i^{n-1}(M \setminus \{g\}) \geq MMS_i^n(M)$.*

Proof. Consider an MMS partition $(M_1, \ldots, M_k) \in \Pi_n(M)$ for agent i. Suppose without loss of generality that $g \in M_n$. By redistributing the items $M_n \setminus \{g\}$ and only selecting the bundles M'_1 to M'_{n-1} we see that $\text{MMS}_i^{n-1}(M \setminus \{g\}) \geq \min_{k \in [n-1]} v_i(M'_k) \geq \min_{k \in [n]} v_i(M_k) = \text{MMS}_i^n(M)$. □

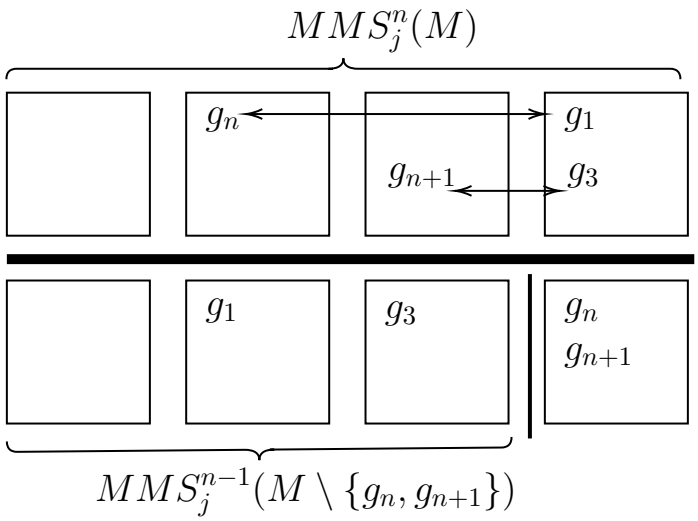


Figure 8: Visual proof of the $\{g_n, g_{n+1}\}$ reduction.

This idea introduced the notion of reductions - we remove one agent and a set of goods and solve the problem on the new instance. As long as other agents' MMS values haven't decreased, any guarantee achieved in the reduced instance is also attained in the original instance. This idea was defined by Ghodsi et al. [2018] as follows.

Definition 9. *The act of removing a set $A_i \subseteq M$ of items and an agent i is a valid **reduction** if the two following conditions hold:*

(i) $v_i(A_i) \geq \beta MMS_i^n(M)$

(ii) $\forall j \in N \setminus \{i\}, MMS_j^{n-1}(M \setminus A_i) \geq MMS_j^n(M)$.

The following family of reductions generalize Lemma 4. The idea of the proof is demonstrated in Figure 8 for $k = 1$.

Lemma 5. *Let $I = \langle N, M, V \rangle$ be an ordered instance. For each $0 \leq k < \frac{m}{n}$, let $S_k = \{g_{kn-k+1}, \ldots, g_{kn+1}\}$. If $v_i(S_k) \geq \beta MMS_i^n$, then giving agent i the bundle S_k forms a valid reduction.*

Proof. By the pigeonhole principle, of the first $kn+1$ goods, at least $k+1$ of them must fall into the same bundle of any $\text{MMS}_j^n(M)$ partition, say bundle A_r. For each good $g \in S_k$ that is not in A_r, it must be in some other bundle, say A_s. Since A_r contains $k+1$ of the first $kn+1$ goods, and it doesn't contain g, it must contain some other good $\hat{g}$ of the first $kn+1$ goods. We swap g and $\hat{g}$ from bundles A_s and A_r respectively. We know that $v_j(g) \leq v_j(\hat{g})$ because S_k contains the last $k+1$ goods of the first $kn+1$ and the instance is ordered. Thus after swapping $v_j(A'_s) = v_j(A_s) + v_j(\hat{g}) - v_j(g) \geq v_j(A_s)$. By continuing these swaps and then redistributing any remaining goods of $A_r \setminus S_k$, the resulting partition satisfies $v_j(A'_t) \geq v_j(A_t)$ for each $t \neq r$. Thus after removing bundles $A_r = S_k$ and agent i, agent j forms an $\text{MMS}_j^{n-1}(M \setminus A_r)$ partition where the minimum value bundle is at least as valuable as $\text{MMS}_j^n(M)$. Thus $\text{MMS}_j^{n-1}(M \setminus A_r) \geq \text{MMS}_j^n(M)$ and giving agent i the set A_r forms a valid reduction. □

Some important results that come out of Lemma 5 are that $\{g_1\}$ and $\{g_n, g_{n+1}\}$ both form valid reductions if any agent desires them. By repeatedly applying these reductions, we may assume that in a reduced instance no agent desires any of these bundles more than their MMS. This implies that no good is worth more than

the MMS and that there are at most n goods worth $\frac{1}{2}$ of the MMS. (This holds because the instance is ordered so $v_i(g_{n+1}) \leq v_i(g_n)$ for all $i \in N$.)

Beyond upper-bounding the value of goods, reductions also allow us to provide lower bounds on the number of goods.

Lemma 6. *In an ordered goods instance, if $m < 2n$ or if $v_i(g_{2n}) = 0$, then giving agent i the good g_1 forms a valid reduction.*

Proof. Suppose that agent i believes that there are less than $2n$ goods. Consider any partition $A = (A_1, \ldots, A_n) \in \Pi_n(M)$. A must include at least one bundle which contains a single item, say item g_k. If g_1 is not in a bundle on its own, suppose that g_1 is in a bundle with g_j. If $v_i(\{g_k, g_j\}) \leq v_i(g_1)$ then observe $v_i(\{g_k, g_j\}) \geq v_i(g_k)$. Otherwise if $v_i(\{g_k, g_j\}) > v_i(g_1)$ then we still have $v_i(g_1) \geq v_i(g_k)$ because the instance is ordered. This implies that $\min(v_i(g_1), v_i(\{g_k, g_j\})) \geq \min(v_i(g_k), v_i(\{g_1, g_j\}))$. Thus we may swap goods g_1 and g_k to form a new partition whose minimum valued bundle is at least as large, and this partition has g_1 in its own bundle. Since this holds for every partition, it must necessarily be the case that a maximin partition for agent i has g_1 in its own bundle. This implies that $v_i(g_1) \geq \text{MMS}_i^n$. By Lemma 4, we know that allocating g_1 to agent i forms a valid reduction. □

Lemma 6 shows that as long as $m < 2n$, we may repeatedly allocate the most valuable good remaining to any agent arbitrarily. Eventually we will either allocate a single good to all agents (if $m \leq n$ at the start) or we will arrive at a smaller instance for which the new $m' = 2n'$.

In the (α, β)-MMS setting, it is important to observe that finding (α, β)-MMS allocations on reduced instances is sufficient for finding (α, β)-MMS allocations on general instances. Whenever $\alpha \leq 1$, $\alpha(n-k)+k = \alpha n - \alpha k + k = \alpha n + (1-\alpha)k \geq \alpha n$. Thus if k agents are allocated bundles during reductions, then an (α, β)-MMS allocation on the reduced instance also gives an (α, β)-MMS allocation on the original instance.

Bag-Filling. Perhaps the most simple algorithm for maximin share approximations is bag-filling. The concept of bag-filling is based upon the idea of the moving knife algorithm for divisible goods [Dubins and Spanier, 1961]. In the moving knife algorithm, we move a knife from left to right along a cake until some agent believes that the cut left of the knife is worth their proportional fair share. Since no other agent shouted before the first agent shouts, the section of the cake taken by any agent is worth at most $\frac{v_i(M)}{n}$ to any other agent. This process repeats until all agents have taken a slice.

When we switch to indivisible goods, the corresponding algorithm is called bag-filling. In the bag-filling algorithm, items are added to a bag until some agent says the bag is satisfactory. If the value of each good is bounded above, we can bound the total value of each allocated bag accordingly.

Lemma 7. *Suppose that $v_i(g) \leq \delta$ for each $i \in N$ and $g \in M$. Then if agents shout for a bundle when it has value Δ, the total value of each allocated bundle is at most $\delta + \Delta$ to each other agent.*

Proof. Consider the last good added to a bag. The bag had value less than Δ to all agents before this good was added (as no agent had shouted previously). The last good has value at most δ, so after the last good is added, the total value of the bundle is at most $\delta + \Delta$. □

In the classical β-approximation of MMS, bag-filling gives a simple algorithm for $(1, \frac{2}{3})$-MMS. This proof was first presented by Garg et al. [2018].

Proposition 1 (Garg et al. [2018])**.** *$(1, \frac{2}{3})$-MMS exists and can be computed in polynomial time.*

Algorithm 4.3: Bag-filling for Goods

Input : An ordered goods instance $\mathcal{I} = \langle N, M, V \rangle$, a threshold Δ
Output: An allocation A

Let $M' = M$, $N' = N$, and $A = (\emptyset, \ldots, \emptyset)$;
for $k = 1$ **to** n **do**
  $B = \emptyset$;
  for $j = 1$ **to** $|M'|$ **do**
    if $\exists i \in N', v_i(B \cup g_j) \geq \Delta$ **then**
      $B = B \cup g_j$;
  $M' = M' \setminus B$;
  Let $i \in N'$ be an agent such that $v_i(B) \geq \Delta$;
  $A_i = B$ and $N' = N' \setminus i$;
return A;

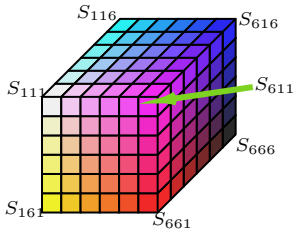
Proof. We claim that Algorithm 4.4 computes a $(1, \frac{2}{3})$-MMS allocation. We first order and scale the instance so that $v_i(M) = n$ for all agents. Using the fact that $\text{MMS}_i^n \leq \frac{v_i(M)}{n} = 1$ (Lemma 14, and by repeatedly applying the reductions from Lemma 5 for $k = 0$ and $k = 1$, we may assume that there are no goods worth more than $\frac{2}{3}$. Likewise, we may assume that there are at most n goods worth $\frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$ or more. If we initialize A_1 to $A_{n'}$ to contain the goods g_1 to g_n respectively, the remaining items all have value less than $\frac{1}{3}$. By running Algorithm 4.3 with $\Delta = \frac{2}{3}$, Lemma 7 implies that each bundle is allocated with total value at most $\frac{1}{3} + \frac{2}{3} = 1$. Thus after $n - 1$ such bundles are allocated, there is still $v_i(M) - (n - 1) = 1$ value remaining, and the last agent may take a bundle worth at least $\frac{2}{3}\text{MMS}_i^n$. Unordering the resulting allocation gives all agents a bundle worth at least as much as in the ordered instance (Lemma 3), so all agents receive a bundle worth at least $\frac{2}{3} \geq \frac{2}{3}\text{MMS}_i^n$. □

Algorithm 4.4: Computing $(1, \frac{2}{3})$-MMS

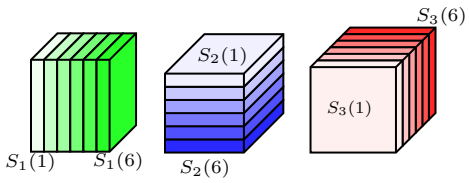
Input : An instance $I = \langle N, M, V \rangle$
Output: A $(1, \frac{2}{3})$-MMS allocation A

1 $I' =$ `Order`(I); `// Algorithm 4.1`
2 $I' =$ `Scale`(I', n); `// Lemma 2`
3 **while** $\exists i \in N, S \in (\{g_1\}, \{g_n, g_{n+1}\}), v_i(S) \geq \frac{2}{3}$ **do**
4 Allocate S to i;
5 $N' = N' \setminus \{i\}$, and $M' = M' \setminus S$;
6 $I' =$`Scale`(I', n');
7 Initialize bags A_1 to $A_{n'}$ with goods g_1 to $g_{n'}$;
8 $A' =$ `Bag-fill`$(I', \frac{2}{3})$; `// Algorithm 4.3`
9 $A =$ `Unorder` (A', I); `// Algorithm 4.2`

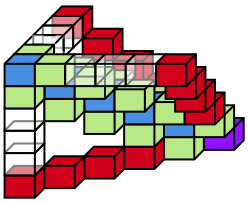
The normalization steps on lines 1-6 and the unorder step at the end of Algorithm 4.4 are standard techniques for computing approximate MMS allocations. While some results require different scaling or reductions, the remaining structure is identical. We first order the instance, scale valuations to make MMS bounds convenient, and apply a set of necessary reductions. After this, the resulting reduced instances have some useful normalized form for which we may a more complicated algorithm. We then unorder the resulting allocation yielding an allocation for the original instance which satisfies (α, β)-MMS as long as the allocation on the reduced instance satisfies (α, β)-MMS.



(9a) The shape of tensor S.



(9b) The slices of tensor S.



(9c) The shape of tensors S and T.

Figure 9: The shape and slices of order 3 tensors with dimensions $(6 \times 6 \times 6)$.

4.2 Optimal-MMS: Understanding the Trade-offs

An optimal-MMS allocation always exists; however, for instances where MMS allocations do not exist, such allocations may leave many agents unsatisfied. In fact, we prove that there exists a family of instances where any optimal-MMS allocation only gives a small constant number of agents their MMS guarantee. Therefore, any optimal-MMS algorithm (e.g. Nguyen et al. [2017]) can perform arbitrarily poorly with respect to satisfying MMS guarantees for a population of agents.

Theorem 1. *For every $n \geq 4$, there exists an instance with $O(n^2)$ goods where every optimal-MMS allocation guarantees at most 3 (4 if n is odd) agents their MMS value.*

The proof is by a construction inspired by those proposed by Kurokawa et al. [2016], but it includes a few intricate modifications. We illustrate our construction for the smallest non-trivial instance: when $n = 6$. The full proof of the theorem can be found in the Appendix B.

Our construction is broken into three components: a tensor S, a tensor T, and 3 groups of agents. The tensors are used to create highly structured preferences for agents. The entries of tensor S are chosen such that equi-partitions of the entries correspond to a set of slices of S. An equi-partition is a partition of the entries of a tensor where each set of the partition has the same sum. Here, all equi-partitions refer to n-partitions. Tensor T provides perturbation so that the only equi-partitions of $S + T$ are aligned slices.

All tensors in our construction are order 3 with dimensions $(6 \times 6 \times 6)$.[7] We say that tensor S is indexed by a 3-tuple (x_1, x_2, x_3) corresponding to entry $S[x_1, x_2, x_3]$ or $S_{x_1x_2x_3}$ for short. The 2-order slices of tensor S along dimension j are given by $S_j(1)$ to $S_j(n)$ where $S_j(i) = \{S_{x_1x_2x_3} \mid x_j = i\}$. We also define $S_{i;x_j=k}$ to be the entry where all indices are i except index $x_j = k$. For example, $S_{i;x_1=n} = S_{nii\ldots i}$. Tensor addition is entry-wise $((S + T)_{x_1,x_2,\ldots,x_d} = S_{x_1,x_2,\ldots,x_d} + T_{x_1,x_2,\ldots,x_d})$. Unless otherwise specified, the entries of a tensor have value 0.

Figure 9 illustrates the shape of a tensor, describes slices of the tensor, and depicts the indexing of the non-zero entries of S and T when there are 6 agents and $d = 3$.

Tensor S. Figure 10 shows the values of tensor S in our construction. The entries $\frac{242}{243}, \frac{80}{81}, \ldots, \frac{365}{486}$ make up the main diagonal (seen in blue on Figure 9c). The first thing to observe is that each slice has total sum 1; this includes the frontal, horizontal, and depth-wise slices. We next observe that the entries on the main diagonal each have value greater than $\frac{1}{2}$, so any equi-partition of the non-zero entries of S must separate these entries. Furthermore, observe that each entry off the main diagonal gets progressively larger. Thus

[7] When referring to a tensor, the "order" is the number of dimensions that the tensor is embedded in. The "dimensions" of a tensor, however, refers to the number of entries along each axis, similar to the dimensions of a matrix.

the only entries that can be allocated with S_{111} are those with value $\frac{1}{486}$ or else there will not be sufficient value for a later set. Starting from the first slice, we see that exactly two entries with these smaller values in the same slice must be allocated in the same set of any equi-partition of S.

While a set of aligned slices forms an equi-partition, it is not necessarily the case that an equi-partition of S must contain only aligned slices. Consider the set of slices $\{S_1(1), S_2(2), S_3(3), S_1(4), S_2(5), S_3(6)\}$. These slices, while not aligned, form an equi-partition of the entries of S. To fix this, we introduce tensor T.

$\frac{242}{243}$					$\frac{1}{486}$												
							$\frac{80}{81}$				$\frac{1}{162}$						
														$\frac{26}{27}$			$\frac{1}{54}$
$\frac{1}{486}$							$\frac{1}{162}$							$\frac{1}{54}$			
												$\frac{1}{486}$					
													$\frac{1}{162}$				
														$\frac{1}{54}$			
			$\frac{8}{9}$		$\frac{1}{18}$										$\frac{1}{18}$		
										$\frac{2}{3}$	$\frac{1}{6}$					$\frac{1}{6}$	
			$\frac{1}{18}$							$\frac{1}{6}$							$\frac{365}{486}$

Figure 10: The values of S where each section of the table corresponds to the slices $S_3(1)$ to $S_3(6)$ from left to right. Empty cells indicate 0. The entries in the first row of the table are S_{111} and S_{611}. The second entry in the first column is entry S_{161}.

Tensor T. Figure 11 shows the values of tensor T in our construction. Observe that each entry in T has value proportional to ϵ^k for $k \geq 1$. The largest entry of T is the entry with value z which is approximately ϵ. We note that while T has negative entries, these values occur where S is positive. Thus if ϵ is sufficiently small, all entries of $S + T$ are non-negative. The entries of T are carefully crafted so that every slice has sum 0. There are three entries (u_i, v_i, w_i) adjacent to each entry on the main diagonal. Any slice $S_j(i)$ will only contain two of these three entries, and the remaining entry will be in the next slice $S_j(i+1)$. The recursive definitions of u_i, v_i, and w_i are chosen with specific cancellations in mind. To illustrate, consider $u_2 + v_2 = r_2 + \frac{u_1 - v_1 - w_1}{2} + c_2 + \frac{-u_1 + v_1 - w_1}{2} = r_2 + c_2 - w_1$. These cancellations occur on each slice so that $T_j(i) = 0$ for each i, j.

We now consider the tensor $S + T$. Since the values of T are all less than ϵ in magnitude, the values of S dominate any partition. Consider the first entry S_{111}. Since two entries from $\{S_{116}, S_{161}, S_{611}\}$ must be put with S_{111} to form an equi-partition of S, the two chosen entries will uniquely identify a slice $(S + T)_j(1)$. Since these entries of S correspond to entries where T has values $-r_1$, $-c_1$, and $-d_1$ respectively, the two corresponding entries of u_1, v_1, and w_1 must be chosen to make this set have total value of 1. Whichever entry is not chosen is left for the next slice. As noted, this small perturbation in the next slice will uniquely select a set which has value 1. Carrying this argument forward yields that the only equi-partitions of $S + T$ are those which correspond to aligned slices.

Grouping Agents. We now group our 6 agents into 3 groups of 2 agents each. For each group, we will enforce that they have a unique set of slices which corresponds to an equi-partition. This is accomplished by adding the perturbation matrix E, seen in Figure 12, to the last slice of each respective set of slices.

	u_1				$-r_1$	w_1											
v_1								u_2			$-r_2$		w_2				
							v_2								u_3		$-r_3$
														v_3			
$-c_1$							$-c_2$							$-c_3$			
												d_1					
													d_2				
		w_3												d_3			
				u_4	$-r_4$				w_4						d_4		
			v_4								x					z	
			$-c_4$							y							a

where:

$$r_i = \epsilon^{15-3i} \qquad c_i = \epsilon^{14-3i} \qquad d_i = \epsilon^{13-3i}$$

$$u_1 = r_1 \qquad v_1 = c_1 \qquad w_1 = d_1$$

$$u_i = r_i + \frac{u_{i-1} - v_{i-1} - w_{i-1}}{2} \qquad v_i = c_i + \frac{-u_{i-1} + v_{i-1} - w_{i-1}}{2} \qquad w_i = d_i + \frac{-u_{i-1} - v_{i-1} + w_{i-1}}{2}$$

$$x = \frac{u_4 - v_4 - w_4}{2} \qquad y = \frac{-u_4 + v_4 - w_4}{2} \qquad z = \frac{-u_4 - v_4 + w_4}{2}$$

$$a = -d_1 - d_2 - d_3 - d_4 - z$$

Figure 11: The values of T. Empty cells indicate 0. The key point to note is that each slice $T_3(i)$ contains the value w_{i-1}. Likewise, the slices $T_1(i)$ contain the value u_{i-1} and the slices $T_2(i)$ contain v_{i-1}.

$-\tilde{\epsilon}$					
	$-\tilde{\epsilon}$				
		$-\tilde{\epsilon}$			
			$-\tilde{\epsilon}$		
				$-\tilde{\epsilon}$	
					$5\tilde{\epsilon}$

Figure 12: The perturbation matrix (E) that separates groups P, Q, and R. An empty cell represents 0.

Consider group 1 where we add E to $(S+T)_1(6)$. Since the total sum of entries in E is 0, group 1 still has an equi-partition with the slices $\{(S+T)_1(1), \ldots, (S+T)_1(6)\}$. However, each entry of E with negative value lies in a different slice in the other two partitions. Thus all but the last slice in these partitions have total value $1-\tilde{\epsilon}$. We note that the entries with value $-\tilde{\epsilon}$ correspond to entries of $(S+T)$ with positive value. Thus if $\tilde{\epsilon}$ is sufficiently small, all entries of the resulting tensor are non-negative.

We now create an instance $I = \langle N, M, V \rangle$ where the goods of M are the non-zero entries of tensor $S+T$ and each agent's valuations are defined by the tensor assigned to their group. Since each agent can form an equi-partition of the values, we have $\text{MMS}_i^6 = 1$ for all agents $i \in N$. By allocating goods according to any set of slices, we see that each agent receives at least $1-\tilde{\epsilon}$ of their MMS. Thus the optimal-MMS ratio is at least $1-\tilde{\epsilon}$. However, since $\tilde{\epsilon}$ is very small compared to the values of S and T, the only way to give each agent at least $1-\tilde{\epsilon}$ value is to give them a slice of the tensor. However, since the slices have to be aligned, and only one group of agents will accept any slice from a given partition, we see that any optimal-MMS allocation gives at most one group (two agents) and a lucky agent who receives the last slice (with perturbation $5\tilde{\epsilon}$) their full MMS.

Generalizing to higher n. By continuing the patterns with valuations, we may extend our order 3 tensors to dimensions $(n \times n \times n)$. Thus by splitting agents into three groups, we may limit optimal-MMS allocations to approximately $\frac{1}{3}$ of the agents who receive their MMS. If we wish to keep the number of agents constant as n grows larger, we need to partition agents into more groups. This can be achieved by embedding our tensor into higher dimensions. In the full proof, we create d groups which leads to an order d tensor. This leads to the more general result that with $O(dn)$ goods, we can guarantee that only $\lceil \frac{n}{d} \rceil + 1$ agents receive their MMS in any optimal-MMS allocation. By selecting $d = \lfloor \frac{n}{2} \rfloor$, we obtain a full proof of Theorem 1.

4.3 (α, β)-MMS for Small n

The previous section demonstrated that maximizing the value of β might minimize the value of α. With this strong counter-example to optimize α and β simultaneously, we ask how slightly weakening α affects the approximation ratio β.

The smallest relaxation of α, where $\alpha = \frac{n-1}{n}$, aims to satisfy all but one agent. Intuitively, by sacrificing one agent, we may redistribute all goods that would have been allocated to them. When n is small, this redistribution will have a greater impact on other agents' bundles. For small values of n, we show that we can outperform the $\frac{3}{4}$ benchmark on all but one agent. Our analysis relies on a modified envy graph algorithm proposed by Barman and Krishna Murthy [2017].

The envy graph algorithm as it was originally proposed by Lipton et al. [2004] guarantees EF1 through a process known as envy cycle elimination. The algorithm operates on an envy graph $G(A) = (V, E)$. The vertices of $G(A)$ correspond to the agents of a given instance, and an edge (i, j) is in E if agent i envies agent j ($v_i(A_i) \leq v_i(A_j)$). The algorithm proceeds in two phases: allocating a good and resolving cycles. Suppose first that $G(A)$ is acyclic. Thus $G(A)$ has a vertex with no incoming edges, and there is an agent who is not envied by any other agent. Then we may give that agent a new good without violating EF1 because this good may be removed to eliminate envy. After adding this item, the envy graph may change. If no envy cycle is created, we may continue this argument for the next good. Suppose that an envy cycle exists after adding this good. That is, there is a sequence of agents $(r_1, r_2, \ldots r_k)$ such that r_k envies agent r_1 and for each $1 \leq r_i < k$, agent r_i envies agent r_{i+1}. We may resolve this cycle by passing bundles backwards through the cycle; agent r_i is given the bundle originally given to agent r_{i+1} and agent r_k is given the bundle of agent r_1. Since the valuation of each agent in the cycle strictly increases the sum $\sum_{i \in N} v_i(A_i)$ also strictly increases. This cycle elimination may create new cycles. We may continue to resolve these cycles until the resulting envy graph is acyclic. Since the set of possible partial allocations is finite, and because $\sum_{i \in N} v_i(A_i)$ strictly increases at each step, this process will eventually terminate in an envy graph without cycles. This process then repeats by giving the next good to some agent who is not envied by any other agent until there are no goods remaining.

Barman and Krishna Murthy [2017] observed that no agent envies a bundle more than the last good added to the bundle. Thus if the envy graph algorithm is modified to run only on ordered instances and the goods are allocated in decreasing order of value, the resulting allocation is EFX.

Lemma 8. *The modified envy graph algorithm (Algorithm 4.5) outputs an EFX allocation.*

Proof. At the start of the algorithm, all bundles are empty which is EFX vacuously. Suppose that the start of iteration j, the partial allocation including goods g_1 to g_{j-1} is EFX. Let agent i be allocated good g_j. Since no agent envied agent i before good g_j was allocated, each agent can remove good g_j from A_i to remove envy. Because goods are added in descending order of value, for each $g \in A_i$, $v_k(g) \geq v_k(g_j)$ which implies

Algorithm 4.5: Modified Envy Graph

Input : An ordered goods instance $\mathcal{I} = \langle N, M, V \rangle$
Output: An EFX allocation A

Initialize $A = (\emptyset, \ldots, \emptyset)$;
for $j = 1$ **to** m **do**
  Let i be a vertex in $G(A)$ with no incoming edges;
  $A_i = A_i \cup \{g_j\}$;
  while $G(A)$ *contains an envy cycle* $r_1, \ldots, r_k$ **do**
    For each $1 \leq i < k$, set $A_{r_i} = A_{r_{i+1}}$ and set $A_k = A_1$;

that for each $g \in A_i$, $v_k(A_k) \geq v_k(A_i \setminus \{g_j\}) \geq v_k(A_i \setminus \{g\})$ and EFX is maintained. During envy cycle elimination, the bundles stay the same and each agent's value for the bundle they receive only increases. Let A'_k be the bundle allocated to agent k after envy cycle elimination. Suppose $v_k(A'_k) < v_k(A'_l \setminus \{g\})$ for some g and A'_l. Suppose agent m was the previous owner of bundle A'_l. Since the partial allocation before envy cycle elimination was EFX, this implies that $v_k(A_k) \geq v_k(A_m \setminus \{g\})$, but this implies that $v_k(A'_k) \geq v_k(A_m \setminus \{g\}) = v_k(A'_l \setminus \{g\})$. Thus agent k does not envy agent l by more than the least valuable good. Since this holds for each k and l after envy cycle elimination, the resulting partial allocation is EFX. Thus before good $j+1$ is allocated, the partial allocation is EFX, and after all m goods are allocated, the resulting allocation is EFX. □

We now show that this same modified envy graph algorithm allows us to compute strong β approximations of MMS for small n with $\alpha = \frac{n-1}{n}$.

Lemma 9. *Given an ordered instance such that* $\forall i \in N$,

- $v_i(g_1) < 1$
- $v_i(\{g_n, g_{n+1}\}) < 1$
- $v_i(M) = n$
- $n \geq 4$

the Modified Envy Graph algorithm on any subset of $n-1$ *agents satisfies* $(\frac{n-1}{n}, \frac{1}{2}(\frac{n+2}{n-1}))$*-MMS.*

Proof. Suppose that instance I satisfies the above conditions. Let A be an allocation produced by the Algorithm 4.5 run on any $n-1$ agents of I. When allocating the first $n-1$ goods, there must always be an agent who has not yet received a good by pigeonhole principle. Since the instance is reduced and scaled such that $v_i(g_1) < 1$ and $v_i(M) = n$, $v_i(g_n) > 0$ (or else there would be at least one good with value $v_i(g) \geq \frac{n}{n-1} > 1$). As the instance is ordered, this implies that all agents have positive value for goods g_1 to g_n. Thus any agent who has not yet received any good envies each agent who has already received a good.

Since the envy graph algorithm does not allocate an item to an envied agent, no agent is allocated a second good until all bundles are allocated at least one good. Define S to be the set of agents who receive only one good in the resulting allocation A and let $s = |S|$.

By Lemma 8, A satisfies EFX, i.e., for all $j \in [n-1]$, $v_i(A_i) \geq v_i(A_j \setminus \hat{g_j})$ for any good $\hat{g_j} \in A_j$. Summing over the set of agents who receive at least two goods:

$$((n-1)-s)v_i(A_i) \geq v_i(M \setminus (\bigcup_{j \in S} A_j)) - \sum_{\substack{j \neq i \\ j \notin S}}^{n-1} v_i(\hat{g_j})$$

$$
\begin{aligned}
((n-1)-s)v_i(A_i) &\geq (n-s) - \sum_{\substack{j\neq i \\ j\notin S}}^{n-1} v_i(\hat{g_j}) && v_i(M) = n \text{ and } v_i(\bigcup_{j\in S} A_j) \leq s \\
((n-1)-s)v_i(A_i) &\geq (n-s) - v_i(g_n) - v_i(g_{n+1}) && \text{Bounding } v_i(\hat{j}) \text{ (only one } g_n) \\
&\quad - ((n-1)-1-s-2)v_i(g_{n+1}) && \\
((n-1)-s)v_i(A_i) &\geq (n-s) - 1 - (n-s-4)v_i(g_{n+1}) && v_i(\{g_n, g_{n+1}\}) < 1 \\
((n-1)-s)v_i(A_i) &\geq (n-s) - 1 - \frac{1}{2}(n-s-4) && v_i(g_{n+1}) \leq \frac{1}{2} \\
((n-1)-s)v_i(A_i) &\geq (n-s) - \frac{1}{2}(n-s-2) && 1 = 2 \cdot \frac{1}{2} \\
v_i(A_i) &\geq \frac{n-s}{n-s-1} - \frac{1}{2}(\frac{n-s-2}{n-s-1}) && \text{dividing by } n-s-1 \\
v_i(A_i) &\geq \frac{(2n-2s)-(n-s-2)}{2(n-s-1)} && \text{Common Denominator} \\
v_i(A_i) &\geq \frac{1}{2}(\frac{n-s+2}{n-s-1}) && \text{Subtraction} \\
v_i(A_i) &\geq \frac{1}{2}(\frac{n+2}{n-1}) && \text{By monotonicity of } \frac{n+2}{n-1}
\end{aligned}
$$

The third step follows because $\sum_{\substack{j\neq i \\ j\notin S}}^{n-1} v_i(\hat{g_j})$ contains (n-1) terms excluding s terms for $j \in S$. For each corresponding agent $j \neq i$ and $j \notin S$, the last good allocated $\hat{g_j}$ must have been from among goods g_n onward. However, only one such $\hat{g_j}$ can be good g_n. We separate the goods g_n and g_{n+1} in order to use $v_i(\{g_n, g_{n+1}\})$ to prove the bound. □

Algorithm 4.6: Computing $(\frac{n-1}{n}, \frac{1}{2}(\frac{n+2}{n-1}))$-MMS

Input : An instance $I = \langle N, M, V\rangle$
Output: A $(\frac{n-1}{n}, \frac{1}{2}(\frac{n+2}{n-1}))$-MMS allocation A

1 $I' =$ `Order(`I`)`; `// Algorithm 4.1`
2 $I' =$ `Scale(`I', n`)`; `// Lemma 2`
3 **while** $\exists i \in N, S \in (\{g_1\}, \{g_n, g_{n+1}\}), v_i(S) \geq 1$ **do**
4 Allocate S to i;
5 $N' = N' \setminus \{i\}$, and $M' = M' \setminus S$;
6 $I' =$`Scale(`I', n'`)`;
7 Let $I' = \langle N' \setminus \{i\}, M'V'\rangle$ for some agent $i \in N'$;
8 $A' =$ `Modified Envy Graph(`I'`)` and $A'_i = \emptyset$; `// Algorithm 4.5`
9 $A =$ `Unorder` (A', I); `// Algorithm 4.2`

Theorem 2. *$(\frac{n-1}{n}, \frac{1}{2}(\frac{n+2}{n-1}))$-MMS exists and is polynomial time computable.*

Proof. Consider Algorithm 4.6. Given an input instance $I = \langle N, M, V\rangle$, lines 1 through 6 construct a new instance I' which ordered and reduced such that for all $i \in N'$, $v_i(g_1) < 1$ and $v_i(\{g_{n'}, g_{n'+1}\}) < 1$. Any agent allocated a bundle during these steps receives value at least $v_i(A'_i) \geq 1 \geq \text{MMS}_i^{n'} \geq \text{MMS}_i^n$.

By Lemma 8, line 8 computes an EFX allocation on $n'-1$ agents of I'. Lemma 9 implies that A' satisfies $(\frac{n'-1}{n'}, \frac{1}{2}(\frac{n'+2}{n'-1}))$-MMS on the reduced I'.

Observe that $\frac{n'+2}{n'-1} \geq \frac{n+2}{n-1}$ when $n' \leq n$ and that $\frac{n'-1}{n'}n' + (n-n') = n-1$. Since unordering does not decrease any agents' value, by Lemma 3, the resulting allocation satisfies $(\frac{n-1}{n}, \frac{1}{2}(\frac{n+2}{n-1}))$-MMS on the original

instance I. Lastly, we notice that the choice of $n-1$ agents was arbitrary, so for each subset of $n-1$ agents, we may compute an allocation which satisfies $(\frac{n-1}{n}, \frac{1}{2}(\frac{n+2}{n-1}))$-MMS and thus $(\frac{n-1}{n}, \frac{1}{2}(\frac{n+2}{n-1}))$-MMS exists. □

n	(α, β)-MMS
4	$(3/4, 1)$-MMS
5	$(4/5, 7/8)$-MMS
6	$(5/6, 4/5)$-MMS
7	$(6/7, 3/4)$-MMS

Table 3: Approximation bounds achieved by Algorithm 4.6 for various $n < 8$ and any number of goods.

Table 3 shows the values achieved by Algorithm 4.6 for $n = 3$ to $n = 7$. We observe that for $n < 7$, Algorithm 4.6 gives better than a $\frac{3}{4}$ approximation of MMS^{n+1}. We also note, that by adding a dummy agent to an instance with 3 agents, and then running Algorithm 4.6, we can compute MMS^4 for all three agents.

4.4 $(\frac{1}{2}, 1)$-MMS Exists

One may consider settings where it is more important that a critical subset of agents/tasks are fulfilled than the majority of them be started. For example, in order to prevent graduation delays, senior college students must be guaranteed seats in classes. Likewise, when resources are scarce in a hospital, a subset of critical tasks may maximize the number of surviving patients. In this setting, we may wish to maximize the number of satisfied agents. In the context of envy-freeness or proportionality, no meaningful guarantee can be made with respect to the number of satisfied agents. Consider n agents with identical valuations where there is a single good worth $1-(n-1)\epsilon$ and $n-1$ goods worth ϵ. Any distribution of goods will leave all but one agent unsatisfied. In the maximin share setting, this leads us to the question, for what values of α does $(\alpha, 1)$-MMS exist and when can such allocations be computed efficiently?

Perhaps the most natural starting point is $(\frac{1}{2}, 1)$-MMS. The following lemma shows that simple algorithms attain this notion.

Proposition 2. *If no agent values any good more than $\frac{v_i(M)}{n}$, then any EF1 allocation on any arbitrary half of the agents satisfies $(\frac{1}{2}, 1)$-MMS.*

Proof. Let $A = (A_1, \ldots, A_{\lfloor n/2 \rfloor})$ be an EF1 allocation on half of the agents of a given instance. For each pair of agents $i, j \in [\lfloor n/2 \rfloor]$, $v_i(A_i) \geq v_i(A_j \setminus g_j)$ for some good $g_j \in A_j$. Let g^* be the most valuable good

for agent i. Combining the inequalities for agent i yields:

$$\begin{aligned}
\sum_{j=1}^{\lfloor \frac{n}{2} \rfloor} v_i(A_i) &\geq \sum_{j=1}^{\lfloor \frac{n}{2} \rfloor} v_i(A_k \setminus g_j) \\
\lfloor \frac{n}{2} \rfloor v_i(A_i) &\geq v_i(M) - \lfloor \frac{n}{2} \rfloor v_i(g^*) \\
v_i(A_i) &\geq \frac{v_i(M)}{\lfloor \frac{n}{2} \rfloor} - v_i(g^*) \\
v_i(A_i) &\geq \frac{v_i(M)}{\frac{n}{2}} - v_i(g^*) \\
v_i(A_i) &\geq \frac{2v_i(M)}{n} - v_i(g^*) \\
v_i(A_i) &\geq \frac{2v_i(M)}{n} - \frac{v_i(M)}{n} \\
v_i(A_i) &\geq \frac{v_i(M)}{n} \geq \text{MMS}_i^n
\end{aligned}$$

□

Proposition 2 implies a simple algorithm for $(\frac{1}{2}, 1)$-MMS. Since $\frac{v_i(M)}{n} \geq \text{MMS}_i^n$, we iterate the reduction $\{g_1\}$ (from Lemma 4) until no good is worth more than $\frac{v_i(M)}{n}$. Then we may select any arbitrary subset of $\lfloor \frac{n}{2} \rfloor$ agents and run any EF1 algorithm (e.g. Round-Robin). The resulting allocation will satisfy $(\frac{1}{2}, 1)$-MMS. Since we may do this for each subset of $\lfloor \frac{n}{2} \rfloor$ agents, we conclude that $(\frac{1}{2}, 1)$-MMS exists.

Corollary 1. *$(\frac{1}{2}, 1)$-MMS exists for additive goods instances.*

4.5 $(\frac{2}{3}, 1)$-MMS Exists

While EF1 allocations provide a simple way to compute $\frac{1}{2}$ approximations for many MMS notions, they often do not lend way to better approximations. A key question is whether we can improve α beyond $\frac{1}{2}$ and show the existence of such allocations. Using the ideas from $(1, \frac{2}{3})$-MMS, we show the existence of $(\frac{2}{3}, 1)$-MMS and discuss an algorithm achieving this bound in polynomial-time when $n < 9$ for any number of goods. With the exception of computing MMS^n for each agent, our proof of $(\frac{2}{3}, 1)$-MMS existence is algorithmic and runs in polynomial time.

Our algorithm extends bag-filling to the case where $\delta + \Delta > 1$. In this setting, Lemma 7 still implies that each bundle allocated has total value less than $\delta + \Delta$ to all agents; however, since $\delta + \Delta > 1$, we may no longer guarantee that each agent receives a bundle. The intuition for our algorithm is straightforward: using the reductions $\{g_1\}$ and $\{g_n, g_{n+1}\}$, we guarantee that no good is worth MMS^n and that there are at most n goods worth $\frac{\text{MMS}^n}{2}$. Then by initializing bundles to separate the high-value goods, we may bag-fill where each item added to a bag has value less than $\frac{\text{MMS}^n}{2}$ to all agents. Thus each allocated bundle has value at most $\frac{3}{2}\text{MMS}^n$ to all agents. We may continue to fill bundles while there is still value remaining, and thus we will be able to allocate approximately $\frac{2n}{3}$ bundles since $\frac{v_i(M)}{\frac{3}{2}\text{MMS}^n} \geq \frac{2v_i(M)}{3\frac{v_i(M)}{n}} = \frac{2n}{3}$.

While the algorithm idea is straightforward, if there are more than $\lfloor \frac{2n}{3} \rfloor$ high-value goods then we will eventually have most of the remaining value tied up in high-value goods. We cannot allocate two high-value goods to a single agent without caution as other agents may value that bundle more than $\frac{3}{2}\text{MMS}^n$. To address this problem, we introduce a notion of *preview bundles*. We allow each agent to form preview bundles by running the $(\frac{2}{3}, 1)$-MMS algorithm as if each agent had their own preferences, but we fix the division

Algorithm 4.7: A $(\frac{2}{3}, 1)$-MMS algorithm for $n < 9$

Input : Instance $I = \langle N, M, V \rangle$ on n agents
Output: A $(\frac{2}{3}, 1)$-MMS allocation A

$I' =$ `Order`(I); // Algorithm 4.1
$I' =$ `Scale`(I', n'); // Lemma 2
while $\exists i \in N, S \in (\{g_1\}, \{g_{n'}, g_{n'+1}\}), v_i(S) \geq 1$ **do**
    Allocate S to i;
    $N' = N' \setminus \{i\}$, and $M' = M' \setminus S$;
    $I' =$ `Scale`(I', n');
while $|N'| > 0$ **do**
    Label items in M' $g_1, g_2, \ldots, g_m$ respectively;
    Let $H = \{g \in M' | \exists i \in N', v_i(g) \geq \frac{1}{2}\}$, let $h = |H|$, and let $L = M' \setminus H$;
    Let $B = \{g_1\}$;
    if $\exists j \in N', v_j(B \cup L) \geq 1$ **then**
        Add goods from L to B in descending order until some agent (say j) values B at least 1;
        Temporarily allocate B to j and remove B from M' and j from N';
    else
        Let $k = 0$;
        while $\exists i \in N'$ *such that* $v_i(g_{h-k}) \geq \frac{1}{2}$ **do**
            Allocate $\{g_{r+k}, g_{h-k}\}$ to i and remove from N';
        if $|N'| > 0$ **then**
            Undo all temporary allocations keeping i out of N';
$A \leftarrow$ `Unorder`(A, I) ; // Algorithm 4.2

of high and low-value goods. Since bag-filling is based on the moving knife protocol, we may make strong comparisons between agents' preview bundles and the bundles allocated by our $(\frac{2}{3}, 1)$-MMS algorithm. This comparison allows us to analyze agents individually in order to show that enough bundles are allocated when bag-filling with the low-value goods. With this intuition, we now define necessary variables to sketch the proof of $(\frac{2}{3}, 1)$-MMS existence.

High/low-value goods. Given an ordered and scaled instance such that $\forall i \in N, \text{MMS}_i \leq 1$, a good $g \in M$ is said to be high-value if there is an agent $i \in N'$ such that $v_i(g) \geq \frac{1}{2}$, otherwise it is low-value. The set of high-value goods is denoted by $H = \{g \in M \mid \exists i \in N', v_i(g) \geq \frac{1}{2}\}$ with $h = |H|$. Additionally, we define the number of surplus high-value goods to be $s = \max(h - \lfloor \frac{2n}{3} \rfloor, 0)$.

Theorem 3. *$(\frac{2}{3}, 1)$-MMS exists and can be computed in polynomial time with an oracle for MMS.*

Proof sketch. We begin by selecting a subset of agents $N' \subseteq N$ where $|N'| \subseteq \lfloor \frac{2n}{3} \rfloor$. We then order the instance and scale valuations so that $v_i(M) = n\text{MMS}_i^n$. We apply the reductions $\{g_1\}$ and $\{g_n, g_{n+1}\}$ only on agents of N' to guarantee that among N', no agent values a good $\text{MMS}_i^n = 1$ or more and that there are at most n goods worth at least $\frac{\text{MMS}_i^n}{2} = \frac{1}{2}$. Observe that this sequence of scaling and reductions is different than in other (α, β)-MMS approximations. While agents in $N \setminus N'$ may have valid reductions, our goal is to guarantee that all agents in N' are guaranteed MMS_i^n. Since our choice of N' is arbitrary, the agents not in N' will be handled by a different choice of N'. Since this proof holds for any choice of N', it follows that $(\frac{2}{3}, 1)$-MMS exists. We also note that the requirement of scaling valuations so that $\text{MMS}_i^n = 1$ for all agents requires knowing agents' MMS values. While such scaling can always be done, it cannot be done in polynomial time unless P = NP since computing MMS values is NP-Hard.

We now begin a modified version of bag-filling that initializes each bundle with the most valuable good

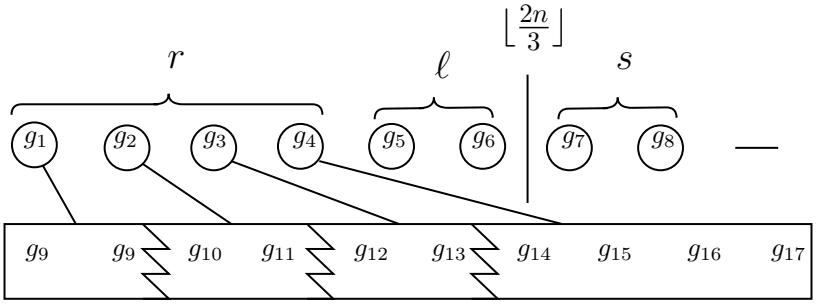


Figure 13: The low-value goods g_9 to g_{17} use a bag-filling procedure to be added to bundles initialized with a single high-value good (g_1 to g_8 each). The bag-filling algorithm fills r bundles with ℓ left to fill when it runs out of low-value goods. The remaining value is tied up with the surplus s high-value goods.

before bag-filling with only low-value goods. If there are high-value goods remaining, then each bundle will have exactly one high-value good. A bundle is then allocated to the first agent in N', say agent i, for which it is worth at least $\text{MMS}_i^n = 1$ in value. Since we only ever add at most one high-value good worth less than 1, every subsequent good has value at most $\frac{1}{2}$. By Lemma 7, we conclude that each allocated bundle has value at most $\frac{3}{2}$ to other agents. After an agent is allocated a bundle during bag-filling, we remove that agent and their allocated bundle from N' and M' respectively. We note that this may update the set of high goods H as it is possible that the removed agent was the only agent who valued g_h at least $\frac{1}{2}$. Any such good that was previously a high-value good is now fair game as a low-value good to other agents.

Suppose that this process stops after r bundles have been allocated. Let $\ell = \lfloor \frac{2n}{3} \rfloor - r$ be the number of bundles left to be allocated (see Figure 13). The total amount of remaining value to each agent in N' who has not received a bundle is at least $n - \frac{3}{2}r = n - \frac{3}{2}(\lfloor \frac{2n}{3} \rfloor - \ell) = n - \frac{3}{2}\lfloor \frac{2n}{3} \rfloor + \frac{3}{2}\ell \geq \frac{3}{2}\ell$. While this remaining value comes from low-value goods, we may continue to fill bundles. The challenge arises when the remaining value is mostly comprised of high-value goods. We analyze cases based upon the number of surplus high value goods s.

Case I: The simplest case is when $s = 0$. In this case, the number of high-value goods is at most $\lfloor \frac{2n}{3} \rfloor$ implying that the remaining value of high-value goods is at most ℓ (each high-value good has value less than 1 and there are at most $\lfloor \frac{2n}{3} \rfloor - r = \ell$ of them remaining). Thus the remaining value from low-value goods is at least $\frac{3}{2}\ell - \ell = \frac{\ell}{2} \geq \frac{1}{2}$. Thus if there are at least $r + 1$ high-value goods, then another bundle can be filled with value at least 1 contradicting that bag-filling stopped. If $h \leq r$, however, then all remaining value comes from low-value goods and another bundle can be filled with only low-value goods, again contradicting that bag-filling stopped. Since this holds for each $r \leq \lfloor \frac{2n}{3} \rfloor$ and s doesn't change, it must be the case that each agent in N' will be allocated a bundle worth at least their MMS_i^n.

Case II: When $\ell > s$, we show by contradiction that bag-filling has not yet stopped. Let $i \in N'$ be an agent who has not yet received a bundle when bag-filling stopped. Consider the preview bundles made by agent i with the set H fixed. We claim that the set of goods in the first k bundles of bag-filling is a subset of the set of goods that agent i puts into their first k preview bundles. Intuitively, since bundles are filled in the same order, we only have to consider when other agents shout before agent i. If other agents would shout after agent i, then because agent i shouts first, their preview bundle and the bag-filling bundle will be the same. Since goods are added to bundles in descending order of value, the number of items added before agent i would shout during bag-filling is at most the number of items in agent i's second preview bundle. Induction on this argument proves the claim. Thus if agent i is able to form at least $\lfloor \frac{2n}{3} \rfloor - s$ preview

bundles, then it must be the case that at least $\lfloor \frac{2n}{3} \rfloor - s$ bundles were allocated during bag-filling (or else agent i would still be able to claim a bundle from the unallocated low-value goods). However, this would imply that $\ell < s$ contradicting that bag-filling had stopped when $\ell > s$. The main technical challenge in our proof is showing that each agent is able to form at least $2(n-h)+s$ preview bundles given a fixed set of high-value goods H.[8] This proves the result as $r \geq 2(n-h)+s$ implies $\lfloor \frac{2n}{3} \rfloor - \ell \geq 2n - 2(\lfloor \frac{2n}{3} \rfloor + s) + s$ which yields $s \geq \ell + 2n - 3\lfloor \frac{2n}{3} \rfloor \geq \ell$.

Case III: When $\ell \leq s$, we observe that there are at least 2ℓ high-value goods remaining. Since we remove agents from N' after allocating them a bundle from bag-filling, we know by the definition of high-value goods that some remaining agent must value g_h at least $\frac{1}{2}$. We may give that agent goods g_{r+1} and g_h as a bundle worth at least 1. Any agent who believes both of these goods are worth at least $\frac{1}{2}$ also agrees that there are $\ell + s$ high-value goods and may continue to pair high-value goods g_{r+k} and g_{h-k}. If less than ℓ bundles can be formed by pairing high-value goods this way, then all remaining agents believe that the last $k+1$ high-value goods are worth less than $\frac{1}{2}$. Thus any bundle allocated by pairing high-value goods is worth at most $\frac{3}{2}$ for these agents. By updating the sets of high-value and low-value goods, these agents are able to fill more bundles during bag-filling. Thus we undo all bundles allocated during the bag-filling steps, but we keep any bundles allocated by pairing high-value goods. We now start a new iteration of bag-filling with the additional low-value goods. We observe that any bundle allocated during bag-filling that decreased $|H|$ will still be allocated in the same order as $|H|$ decreases further after we hit the point where pairing occurred. Since the set of permanent allocations strictly increases in size at each iteration, this process will eventually terminate with an allocation where at least $\lfloor \frac{2n}{3} \rfloor$ agents receive a bundle worth at least 1.

□

We observe that the assumption that $\text{MMS}_i = 1$ for all agents cannot be satisfied in polynomial time. One natural assumption would be that by using the PTAS of Woeginger [1997], we could compute $(\frac{2}{3}, 1-\epsilon)$-MMS. In practice, however, this weakening may sometimes yield allocations which do not satisfy $(\frac{2}{3}, 1-\epsilon)$-MMS.

We construct a family of instances with $n \geq 9$ in which a small error in computing the MMS bound causes Algorithm 4.7 to stop before $\lfloor \frac{2n}{3} \rfloor$ bundles have been allocated. The challenge presented here is not unique to our algorithm. Any algorithm which satisfies $(\frac{2}{3}, 1)$-MMS must correctly determine that $\text{MMS}_i^n = 1 - \epsilon$ for all agents. However, this task is NP-hard even when agents have identical valuations [Woeginger, 1997].

Remark 1. *Consider $n \geq 9$ agents with identical valuations as follows: $n-1$ high-value goods of value $1-\epsilon-\tilde{\epsilon}$, two goods of value $\frac{1}{2}-\epsilon$, one good of value $(n+1)\epsilon$, and $n-1$ goods of value $\tilde{\epsilon}$, where $\epsilon \gg \tilde{\epsilon}$. Then, $\text{MMS}_i^n = 1-\epsilon$ for all agents.*

Algorithm 4.7 allocates three bundles during the bag-filling steps, accounting for the three goods of value $1-\epsilon-\tilde{\epsilon}$, both goods of value $\frac{1}{2}-\epsilon$, and the good with value $(n+1)\epsilon$. After bag-filling, the remaining high-value goods will be paired together to create a total of $\lfloor \frac{n-4}{2} \rfloor$ bundles of paired high-value goods. Thus, Algorithm 4.7 only satisfies $\lfloor \frac{n}{2} \rfloor + 1$ agents.

This surprising remark shows that an inexact MMS bound limits the guarantee of Algorithm 4.7 to at most $(\frac{1}{2}, 1)$-MMS. Fortunately, we are able to show that Algorithm 4.7 in fact computes an $(\frac{2}{3}, 1)$-MMS allocation in polynomial time when $n < 9$, even without access to an MMS oracle.

Theorem 4. *For $n < 9$, Algorithm 4.7 computes a $(\frac{2}{3}, 1)$-MMS allocation in polynomial time.*

[8] A proof of this claim can be found in Appendix C.

Proof sketch. We assume bag-filling stopped after $r = \lfloor \frac{2n}{3} \rfloor - \ell$ bundles and that there were $s = \max(h - \lfloor \frac{2n}{3} \rfloor, 0)$ surplus high-value goods. Again, we consider cases based on s and ℓ. The proof for the cases where $s = 0$ and $\ell \leq s$ are identical to the cases in Theorem 3. When $\ell \geq 2s$ the amount of value remaining from low-value goods is at least $\frac{3}{2}\ell - (\ell + s)v_i(g_{r+1}) \geq \frac{3}{2}\ell - (\ell + \frac{\ell}{2})v_i(g_{r+1}) = \ell(\frac{3}{2} - v_i(g_{r+1}) \geq 1 - v_i(g_{r+1})$ which implies that another bundle may be filled. This contradicts that bag-filling stopped. We show by explicit enumeration that if $n < 9$, then the cases where $s < \ell < 2s$ do not occur when $v_i(M) = n$ for all agents.

All above steps, including scaling agents' valuation such that $v_i(M) = n$ and the steps of the bag-filling algorithm and its iterations, only require polynomial-time computation. Thus, computing $\text{MMS}^{\frac{2}{3}}1$ allocations for $n < 9$ is in polynomial time. □

The analysis in Theorem 3 and Theorem 4 are indifferent to the set of agents available in N'. Consequently, we may select which subset of $\lfloor \frac{2n}{3} \rfloor$ agents are given their full MMS. We also observe that the instance given in Remark 1 is tight; when $n < 9$, $\lfloor \frac{n}{2} \rfloor + 1 \geq \lfloor \frac{2n}{3} \rfloor$. Furthermore, combining this construction with case 3 of Theorem 4 implies that no smaller counter-example exists. We note that despite this bound, the algorithm is still practical as there is no bound on the number of goods. For example, more than 99% of instances in the *Spliddit* dataset deal with less than 9 agents.

4.6 Relationships Between MMS Approximations

We begin with a simple lemma which demonstrates the relationship between βMMS^n and $\text{MMS}^{\gamma n}$.

Lemma 10. *$q\text{MMS}^{kq} \leq \text{MMS}^k$ for any positive integers k and q.*

Proof. Given kq bundles of an MMS partition, we can form a k partition of the same items by grouping q bundles together k times. The lowest valued bundle of this partition gives a lower bound on MMS^k. Thus $q\text{MMS}^{kq} \leq \text{MMS}^k$. □

We observe that this relationship only holds when q is an integer. Consider $q = \frac{3}{4}$ in the following two instances:

(i) 12 goods with value 1
(ii) 6 goods with value 1; 6 goods with value $\frac{1}{2}$

In the first instance $\frac{3}{4}\text{MMS}^{\frac{3}{4}\cdot 12} = \frac{3}{4} \leq 1 = \text{MMS}^{12}$. In the second instance $\frac{3}{4}\text{MMS}^{\frac{3}{4}\cdot 12} = \frac{3}{4} \geq \frac{1}{2} = \text{MMS}^{12}$. Thus we may neither say that $q\text{MMS}^{kq} \leq \text{MMS}^k$ nor $q\text{MMS}^{kq} \geq \text{MMS}^k$ in general when q is not an integer.

The most important consequence of Lemma 10 is that $\text{MMS}^n \geq 2\text{MMS}^{2n}$. This implies that an algorithm which computes $(1, \frac{1}{2})$-MMS also computes MMS^{2n}. We now show that this can be leveraged to compute a $(\frac{1}{2}, 1)$-MMS allocation.

Lemma 11. *$(\alpha, 1)$-MMS exists if and only if $\text{MMS}^{\lceil \frac{n}{\alpha} \rceil}$ exists.*

Proof. Let $I = \langle N, M, V \rangle$ be an additive instance where $|N| = n$. Add $\lceil \frac{n}{\alpha} - n \rceil$ dummy agents to N and call the new instance $I' = \langle N', M, V \rangle$. Thus, $n' = |N'| = n + \lceil \frac{n}{\alpha} - n \rceil = \lceil \frac{n}{\alpha} \rceil$. $(\alpha, 1)$-MMS existence on I' implies that for each subset N'' of $\alpha n'$ agents, an allocation exists which gives each agent in N'' their $\text{MMS}^{N'}$. Since $\lfloor \alpha n' \rfloor = \lfloor \alpha \lceil \frac{n}{\alpha} \rceil \rfloor \geq \lfloor \alpha(\frac{n}{\alpha}) \rfloor = \lfloor n \rfloor = n$ we may select the original subset $N \subseteq N'$. Thus there exists an allocation $A = (A_1, \ldots A_n)$ such that for all $i \in N, v_i(A_i) \geq \text{MMS}_i^{|N'|}$. Replacing the size of N', we have $\text{MMS}_i^{\frac{n}{\alpha}}$, which implies $\text{MMS}_i^{\lceil \frac{n}{\alpha} \rceil}$ exists for I.

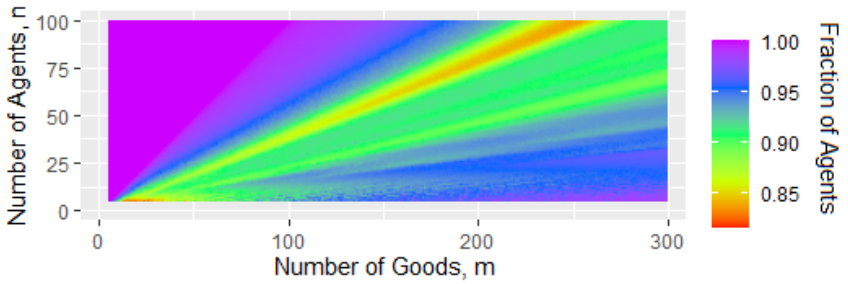


Figure 14: Fraction of agents, α, receiving their maximin share, i.e., $(\alpha, 1)$-MMS.

For the reverse direction, suppose that $\text{MMS}^{\lceil \frac{n}{\alpha} \rceil}$ exists. Consider an additive instance $I = \langle N, M, V \rangle$ with $|N| = n$. Select $n' = \lfloor \alpha n \rfloor$ of these agents arbitrarily as N'. Call the new instance $I' = \langle N', M, V \rangle$. Since $\text{MMS}^{\lceil \frac{n'}{\alpha} \rceil}$ exists, then all agents in N' can be guaranteed at least this value. Replacing n' with $\lfloor \alpha n \rfloor$, we get $\lceil \frac{n'}{\alpha} \rceil = \lceil \frac{1}{\alpha} \lfloor \alpha n \rfloor \rceil \leq \lceil \frac{1}{\alpha} (\alpha n) \rceil = \lceil n \rceil = n$ which implies that $\lfloor \alpha n \rfloor$ agents received $\text{MMS}^{\lceil \frac{n'}{\alpha} \rceil} \geq \text{MMS}^n$ and thus the resulting allocation satisfies $(\alpha, 1)$-MMS on I. □

One consequence of the previous lemma is that any $(1, \frac{1}{2})$-MMS algorithm can also be used to satisfy $(\frac{1}{2}, 1)$-MMS. More importantly, combining Theorem 3 with Lemma 11 shows that $\text{MMS}^{\lfloor \frac{3n}{2} \rfloor}$ exists which significantly improves the previous boundary of MMS^{2n-2} [Aigner-Horev and Segal-Halevi, 2019]. We note that this is the first non-trivial improvement over the MMS^{2n} approximation.

Corollary 2. *Given an additive instance, an allocation satisfying* $\text{MMS}^{\lceil \frac{3n}{2} \rceil}$ *always exists. Moreover, such allocations can be computed in polynomial time when* $n < 6$.

4.7 Empirical Evaluations

We empirically evaluated Algorithm 4.7 both on synthetic and real datasets. In simulated experiments, we focused on ordered instances as they are the most difficult instances in achieving MMS [Bouveret and Lemaître, 2016]. Valuations were sampled uniformly at random, and then ordered and scaled such that $v_i(M) = n$ for all $i \in N$. We generated 1,000 instances for each combination of n and m.

Figure 14 illustrates the outcome of Algorithm 4.7 for $n = 5$ to 100 agents and $m = 5$ to 300 goods. In almost all instances, the $(\frac{2}{3}, 1)$-MMS algorithm goes beyond its $\frac{2}{3}$ bound: more than 88% of agents receive their MMS, with this bound improving as either n or m increases. We observe linear bands where a lower fraction of agents are satisfied due to the ratio of goods to agents. The most prominent of these occurs along the line $m = 3.43n + 2.46$, which was found via linear regression along points where less than 88% of agents receive their MMS. These occurrences account for only 3.64% of total observations.

To paint a more complete picture, we compared the performance of our algorithm with the $(1, \frac{2}{3})$-MMS algorithm of Garg et al. [2018] as shown in Figure 15. We observe that although the $(1, \frac{2}{3})$-MMS algorithm guarantees $\frac{2}{3}\text{MMS}_i$, for all $i \in N$, in almost all instances only a small fraction of agents receive their MMS.

Lastly, we evaluated the performance of Algorithm 4.7 given real fair division instances from *Spliddit* [Goldman and Procaccia, 2014]. Table 4 shows the number of real world instances where our algorithm allocated various fractions of agents their full MMS guarantee. We observe that Algorithm 4.7 satisfied more than $\frac{2}{3}$ of the agents in all 2,395 fair division instances (with $n \geq 3$) and in most instances satisfied all agents (72% of instances). In contrast, the $\text{MMS}1\frac{2}{3}$ approximation algorithm from Garg et al. [2018] only satisfied

45% of the agents. Figure 16 compares the distribution of instances where various fractions of agents receive their MMS.

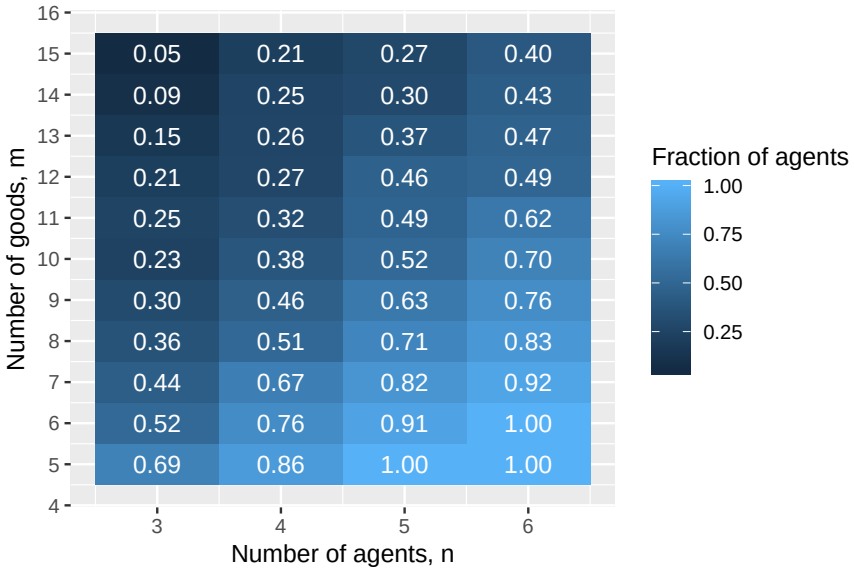


Figure 15: Fraction of agents receiving their MMS under Garg et al.'s MMS1$\frac{2}{3}$ algorithm.

Fraction Satisfied	Garg et al. Algorithm	Algorithm 4.7
1	1,094	1,744
0.75 – 0.83	23	12
0.67	930	639
0.4 – 0.6	14	0
0.33	297	0
0 – 0.33	37	0
Total	2,395	2,395

Table 4: Number of instances where Algorithm 4.7 and the $(1, \frac{2}{3})$-MMS algorithm by Garg et al. [2018] satisfy various fractions of the agents. The instances are from the Spliddit dataset selecting for $n \geq 3$.

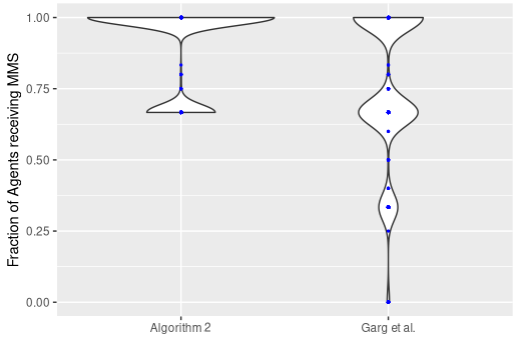


Figure 16: Fraction of agents receiving their MMS under the $(\frac{2}{3}, 1)$-MMS (Alg. 4.7) and $(1, \frac{2}{3})$-MMS (Garg et al. [2018]) algorithms on Spliddit data.

5 Maximin Share for Chores

5.1 A Discussion on Definitions

The definition of Maximin Share is well-defined for chores. Each agent still tries to divide chores into the most favorable partition for themself given that they must select last. Viewing chores as negatively valued items, this definition still leads to

$$\text{MMS}_i^k = \max_{(A_1,\ldots,A_k)\in\Pi_k} \min_{j\in[k]} v_i(A_j).$$

While analysis in this domain is relatively straightforward, it is often helpful to think about chores by the magnitude of their value rather than their exact value. For example, a key aspect in approximating MMS for goods is bounding the best items. Intuitively, these techniques target the items which are least divisible; however, for chores instances, the least divisible items are the worst chores. In light of this intuition, some research on chores operates on positive dis-utilities where each agent's goal is to minimize the amount of dis-utility they receive. In this setting, the goal of maximin share is replaced with a perverse minimax share. That is, each agent requires at most as much dis-utility as the minimum they can guarantee themself by selecting a partition if they are required to pick their worst bundle (the bundle with highest dis-utility).

Definition 10. *The minimax share (mMS) guarantee is the amount of value an agent can guarantee if they select their favorite bundle from their worst partition. Formally*

$$mMS_i^n(M, v_i) = \min_{(A_1,\ldots,A_n)\in\Pi_n(M)} \max_{j\in[n]} v_i(A_j).$$

An allocation $A = (A_1, A_2, \ldots A_n) \in \Pi_n(M)$ is a satisfies the β perverse minimax share criteria if for all $i \in N$, $v_i(A_i) \leq \beta mMS_i^n(M, v_i)$.

This relationship between the maximin share for chores and the perverse minimax share are characterized in the following lemma by Aziz et al. [2017].[9]

Lemma 12. *An allocation $A = (A_1, \ldots, A_n) \in \Pi_n(M)$ satisfies βMMS^n for I if and only if A is a solution of the perverse βmMS problem for the corresponding instance $-I$.*

[9]This lemma is stated as a proposition without proof in Aziz et al. [2017]. We add a proof for completeness.

Proof. We first prove that $-\text{MMS}_i^k(M, v_i) = \text{mMS}_i^k(M, -v_i)$. Consider any partition $(P_1, \ldots, P_k) \in \Pi_k(M)$. When multiplying valuations by -1, the minimum valued bundle becomes the maximum valued bundle. Likewise, over the set of values corresponding to minimum bundles in each partition, multiplying by -1 maps the maximum such value to the minimum such value. Thus:

$$\begin{aligned}
-\text{MMS}_i^k(I) &= -\max_{(P_1,\ldots,P_k)\in\Pi_k} \min_{j\in[k]} v_i(P_j) \\
&= \min_{(P_1,\ldots,P_k)\in\Pi_k} -\min_{j\in[k]} v_i(P_j) \\
&= \min_{(P_1,\ldots,P_k)\in\Pi_k} \max_{j\in[k]} -v_i(P_j) \\
&= \text{mMS}_i^k(-I).
\end{aligned}$$

Furthermore, any maximin partition on I corresponds to a minimax partition on $-I$. Now suppose that $A = (A_1, \ldots, A_k) \in \Pi_k$ satisfy $v_i(A_i) \geq \beta\text{MMS}_i^k(I)$ for all $i \in [k]$. Then $-v_i(A_i) \leq -\beta\text{MMS}_i^k(M, v_i) = \beta\text{mMS}_i^k(M, -v_i)$. Observe that this result is bi-directional implying an equivalence between maximin share allocations and perverse minimax allocations. □

In the context of goods, we have a few fairly intuitive approximations for MMS. The most common approximation is to appease everybody by guaranteeing each agent receives a β fraction of their MMS. Likewise, we might also approximate MMS by telling agents that they must share goods among γn agents where $\gamma \geq 1$. In this approximation each agent expects to receive less value, so we may satisfy their $\text{MMS}^{\gamma n}$. Alternatively, we proposed approximating MMS by giving α fraction of the agents their full MMS value. We showed some interpolations between these notions, and addressed relevant open problems.

In the context of chores, these notions are somewhat less intuitive. Inexact partitions leave agents with less value than their MMS, and thus these agents receive a greater magnitude of work than they were expecting. Thus we may naturally extend β approximations by requiring that each agent receives at most $\beta \geq 1$ times their MMS value. When we consider allocating chores, agents are unsatisfied if they receive too much work. One intuitive way to prevent overburdening agents is to introduce freelancers to the market to reduce the actual workload on each agent. However, an important question arises when we ask how much work the freelancer is allowed to perform. Clearly we may always add a freelancer with zero value for all chores and receive a trivial allocation (the freelancer takes all chores) which satisfies MMS for all agents in the market (including the freelancer).

One possible idea is to have freelancers take valuations based on agents within the market. The simplest way to achieve this is to have freelancers which duplicate agents valuations. Trivially, this gives MMS existence with $n - 1$ freelancers added to the market: create $n - 1$ copies of any agent in the market and allocate bundles to each copy (including the original) according to their MMS partition for n agents.

Another possible extension is to allow freelancers to have valuations which are a convex combination of the valuations of other agents. Either for each chore individually or for each valuation profile. While each of these possibilities are mathematically interesting in their own right, they hint towards a less ambiguous notion which generalizes the γ approximations of goods. Instead of adding agents to the market after agents compute their MMS, we may tell agents to compute their MMS with fewer agents in the market. If agents expect to be understaffed, they will be mentally prepared for a higher workload and thus might be satisfied with allocations which are close to satisfying MMS^n. In this case, we wish to give all n agents in the market at least as much value as their $\text{MMS}^{\gamma n}$ where $\gamma \leq 1$.

In the case of chores, we formalize these approximations with the following definitions.

Definition 11. *Consider an allocation $A = (A_1, \ldots, A_n) \in \Pi_n(M)$.*

- *We say A satisfies $MMS^{\gamma n}$ (where $\gamma \leq 1$) if $v_i(A_i) \geq MMS_i^{\gamma n}$ for each agent $i \in N$.*
- *We say A satisfies βMMS^n (where $\beta \geq 1$) if $v_i(A_i) \geq \beta MMS_i^n$ (Aziz et al. [2017]).*

As in the case of goods, the definitions for chores are related for integer multiples.

Lemma 13. *$qMMS^{kq} \leq MMS^k$ for any positive integers k and q.*

Proof. Given $n = kq$ bundles of an MMS partition, we can form a k partition of the same items by grouping q bundles together k times. The lowest valued bundle of this partition gives a lower bound on MMS^k. Thus $q\text{MMS}^{kq} \leq \text{MMS}^k$. □

Like in the case for goods, this relationship only holds when q is an integer. Consider $q = \frac{3}{4}$ in the following two instances:

(i) 9 chores with value -1; 3 chores with value $-\frac{1}{2}$
(ii) 6 chores with value -1; 6 chores with value $-\frac{1}{2}$

In the first instance $\text{MMS}^{12} = -1 \geq \frac{3}{4} \cdot -\frac{3}{2} = \frac{3}{4}\text{MMS}^9$. In the second instance $\text{MMS}^{12} = -1 \leq \frac{3}{4} \cdot -1 = \frac{3}{4}\text{MMS}^9$. Thus we claim neither $q\text{MMS}^{kq} \leq \text{MMS}^k$ nor $q\text{MMS}^{kq} \geq \text{MMS}^k$ hold in general when q is not an integer.

Lastly, we observe that the notion of $(\alpha, 1)$-MMS is trivial in the context of chores. We may always satisfy all but one agent by giving all chores to a single agent. Since each other agent receives value 0, they are satisfied.

Observation 1. *For a chores instance $I = \langle N, M, V \rangle$, $(\frac{n-1}{n}, 1)$-MMS exists by giving all chores to any agent.*

5.2 Optimal-MMS fails for Chores

We observe that the optimal-MMS counterexample for goods extends to the case of chores with relatively simple modifications. Similar to the case of goods, this shows that we cannot hope to optimize both α and β simultaneously in MMS approximations.

Theorem 5. *For every $n \geq 4$, there exists an instance with $O(n^2)$ chores where every optimal-MMS allocation guarantees at most 3 (4 if n is odd) agents their MMS value.*

The proof for chores is almost identical to the proof for goods.[10] By multiplying $(S + T)$ by -1, the same combinatorial structure holds where each agent is able to form an equi-partition with each set having value -1. Since Prop still implies MMS in the context of chores, $\text{MMS}_i \leq -1$ implying tightness where each agent has $\text{MMS}_i = -1$. When we enforce that the partitions are unique (by adding matrix E), we need to continue to enforce that all but one slice is inadequate in the other partitions. Thus as long as we do not multiply E by -1, the same structure will still hold yielding a similar counterexample.

We observe that Theorem 5 has larger implications beyond optimal-MMS. Prior to this work, Aziz et al. [2017] proved that for three agents, there exists an instance on 12 chores where MMS does not exist. While

[10] The full proof can be found in Appendix D.

this result generalized the construction for $n = 3$ to the case of chores, it was previously not shown that such a counterexample still held for any $n > 3$. We observe that Theorem 5 proves that for any $n \geq 4$, that there exists an instance where MMS does not exist.

Corollary 3. *For any $n \geq 4$, there exists a chores instance $I = \langle N, M, V \rangle$ for which no MMS allocation exists.*

5.3 A Discussion of Translating Standard Lemmas from Goods

In the context of goods, we rely upon the following ideas to build simple algorithms to approximate MMS: scale invariance, proportionality upper bound, ordering instances, reductions, and bag-filling.

Scale Invariance. In chores instances, the maximin share allocation remains independent of scale. Each agent only cares about how they partition goods to compute their maximin share values and scaling all chore values simultaneously leaves each bundle scaled the same amount as the maximin share. The proof of scale invariance for chores is identical to the proof of scale invariance for goods. Because of this, we may often choose to scale instances so that $\text{MMS}_i \geq -1$ for all agents.

Proportionality Upper Bound. For chores the proof that $\text{MMS}_i^k \leq \frac{v_i(M)}{k}$ still holds.

Lemma 14. *$MMS_i^k \leq \frac{v_i(M)}{k}$ for chores instances.*

Proof. Suppose for contradiction that $\text{MMS}_i^k > \frac{v_i(M)}{k}$. Then adding together all bundles of an MMS partition yields $v_i(M) \geq k\text{MMS}_i^k > k\frac{v_i(M)}{k} = v_i(M)$ which is a contradiction. □

Ordering Instances. Independent of item valuations, ordered instances present the most conflict among agents. When agents agree on the order of item valuations, they might compete to receive each item in turn rather than assign each agent their unique favorites before moving on to the next item. Naturally, the relationship between ordered instances and MMS still holds, and the algorithm by [Barman and Krishna Murthy, 2017] still allows us to consider only identical ordinal preferences. However, it is often convenient to define ordered instances for chores so that $|v(b_1)| \geq |v(b_2)| \geq \ldots \geq |v(b_m)|$.

Definition 12. *A chores instance $I = \langle N, M, V \rangle$ is **ordered** if there exists a labeling of the chores such that for all $i \in N$, $|v_i(b_1)| \geq |v_i(b_2)| \geq \ldots \geq |v_i(b_m)|$.*

With this new definition of an ordered instance, we adapt the ordering and unordering algorithms for goods with simple modifications. The corresponding algorithms for chores can be seen in Algorithm 5.1 and Algorithm 5.2.

Lemma 15. *Let $I' = \langle N, M, V' \rangle$ be an ordered chores instance constructed from the original instance $I = \langle N, M, V \rangle$. Given allocation A' on I', a corresponding allocation A on I can be computed in polynomial time such that for all $i \in N, v_i(A_i) \geq v'_i(A'_i)$.*

Proof. Consider Algorithm 5.2. The proof is nearly identical to the proof for goods. We merely observe that the assumption $|v_i(b_1)| \geq |v_i(b_2)| \geq \ldots \geq |v_i(b_m)|$ on chores implies that $v_i(b_m) \geq v_i(b_{m-1}) \geq \ldots \geq v_i(b_1)$. Thus in both ordering and unordering, we reverse the order in which we address items. Suppose agent i received their kth favorite item: $b_{(m+1)-k}$; it is still the case that after unordering the first $k-1$ items, their kth favorite item must still remain unallocated. This implies that agent i is able to take a chore worth at

Algorithm 5.1: Ordering an Instance

Input : Instance $I = \langle N, M, V \rangle$ on n agents
Output: An Ordered instance $I' = \langle N, M, V' \rangle$

for $j = 1$ **to** m **do**
  for $i = 1$ **to** n **do**
    Let $b =$ agent i's jth most valuable item;
    $v'_i(b_{(m+1)-j}) = v_i(b)$;

Algorithm 5.2: Unordering an Allocation

Input : Allocation $A' = (A'_1, \ldots, A'_n)$ for an ordered instance I' of I
Output: Allocation $A = (A_1, \ldots, A_n)$ for the unordered instance I

$A = (\emptyset, \ldots, \emptyset)$ and $M' = M$;
for $j = m$ **to** 1 **do**
  Let agent i be the agent who received b_j in A'_i;
  Let $b = \arg\max_{b \in M'} v_i(b)$;
  $A_i = A_i \cup \{b\}$ and $M' = M' \setminus \{b\}$;

Figure 17: Algorithms for ordering a chores instance and unordering a chores allocation.

least $v_i(b_{(m+1)-k})$ at this step. Since this property holds by induction, and the best item is always available first, we see that each item allocated during unordering is at least as valuable as the corresponding item allocated in the ordered instance. Thus $v_i(A_i) \geq v'_i(A'_i)$ by additivity of items in A_i and A'_i. □

Reductions. The idea of a reduction still holds for chores instances. If there is some bundle S worth at least MMS^n_i to agent i and no other agent's MMS decreases after allocating that bundle ($\text{MMS}^{n-1}_j(M \setminus S) \geq \text{MMS}^n_j(M)$ for all $j \neq i$), then we may allocate that bundle to agent i leaving a reduced instance.

While the definition of reductions still hold, we cannot use the same reductions for chores as we do for goods. The reductions for goods rely upon the fact that redistributing items from one bundle of a partition weakly increases the value of other bundles. However, in the context of chores, this assumption does not hold. In fact, even the simple $\{g_1\}$ reduction does not hold for chores.

Example 1. Consider an instance with 3 agents and 6 chores each with value -0.5. $\text{MMS}^3(M) = -1$ for each agent, but removing a single good and a single agent leaves $\text{MMS}^2(M \setminus \{b\}) = -1.5$.

Likewise, the other reductions for goods not hold for chores. We may generalize the above example for the reduction $\{g_{kn-(k-1)}, \ldots, g_{kn+1}\}$.

Example 2. Consider an instance with 3 agents and $3(k+2)$ chores of value $-\frac{1}{k+2}$. Trivially, each agent has $\text{MMS}^3(M) = -1$, so any a bundle B of $k+1$ chores has value $-\frac{k+1}{k+2} \geq \text{MMS}^3$. However, removing a single good and this bundle leaves $2k+5$ items and $\text{MMS}^2(M \setminus B) = -\frac{k+3}{k+2} < \text{MMS}^3(M)$.

Despite this apparent loss of a critical tool, there is still good news for chores algorithms. We observe that the objective of the reductions for goods is to upper bound the largest magnitude of goods. In the case of chores, we may still use the intuition from goods reductions without actually having the same reductions available. Consider for example b_1, the chore with largest magnitude. Since this chore must be in a bundle of the MMS partition, we know that $\text{MMS}_i \leq v_i(b_1)$. Consequently, we may again say that $|v_i(b_1)| \leq |\text{MMS}_i|$. Likewise, this same relationship with goods holds for each standard reduction.

Lemma 16. *Let $I = \langle N, M, V \rangle$ be an ordered chores instance. For each $0 \leq k < \frac{m}{q}$, there are at most kq chores b such that $|v_i(b)| \geq |\frac{MMS^q_i}{k+1}|$.*

Proof. Consider the last $k+1$ of the first $kq+1$ chores; we know that at least $k+1$ of these must fall within the same bundle of any MMS^q partition (by pigeonhole). The magnitude of these chores are minimized if

the lowest $k+1$ magnitude of these chores fall into the same bundle. Since the lowest value bundle of any MMS^q partition contains at most as much magnitude as these $k+1$ chores, we may say that MMS_i^q is worth at most $v_i(\{b_{kq-(k-1)}, \dots b_{kq+1}\})$. Since the chores are ordered, we know that $v_i(b_{kq+1}) \geq \frac{\text{MMS}_i^q}{k+1}$. In turn, this implies that there are at most kq chores worth less than $\frac{\text{MMS}_i^q}{k+1}$. □

Lemma 16 implies that most of the results derived from reductions for goods still hold for the case of chores. For example, setting $q = n$ and $k = 0$ shows that there are no chores with value $|v_i(b)| \geq |\text{MMS}_i^n|$. Likewise, setting $k = 1$ implies that there at most n goods with value $|v_i(b)| \geq |\frac{\text{MMS}_i^n(M)}{2}|$ as $|v_i(\{b_n, b_{n+1}\})| \leq |\text{MMS}_i^n|$. While this extends most of the reduction for goods, there is still one important result which comes from goods reductions which extends to the case of chores: a lower bound on the number of items.

Lemma 17. *If $m < 2n$, giving $\{b_1\}$ to any agent is a valid reduction.*

Proof. If the total number of items is less than $2n$, there must be some bundle in every MMS partition which contains only a single item. Without loss of generality, we may assume that the worst chore is in a bundle by itself. Since this holds for all agents, and we may assume without loss of generality that the instance is ordered, all agents value b_1 at least as much as their maximin share. Furthermore, removing a single agent and chore b_1 does not change any of the remaining bundles, and thus does not decrease the MMS. This implies that whenever $m < 2n$, there exists a reduction by giving any agent chore b_1. □

By repeatedly applying the reduction from Lemma 17, we can focus our attention on cases where $m \geq 2n$.

Bag-Filling. Bag-filling for goods is inspired by the moving-knife algorithm: we (discretely) move a knife along the set of goods and allocate it whenever it becomes sufficient in value for some agent. By upper bounding the value of the allocated bundle to other agents, we guarantee that there is enough remaining value other agents may still fill bundles as desired. When we consider the equivalent algorithm for chores, the objective is flipped. Since each agent may take an empty bundle, the challenge is to ensure that all chores are allocated. Thus when we fill a bundle, we wish to maximize the number of chores added to the bundle before it is allocated. As long as some agent still finds the bundle acceptable, we may continue to add chores. This idea is formalized in Algorithm 5.3. This algorithm, originally proposed by Huang and Lu [2019], encapsulates this acceptability criterion with threshold values for each agent. Setting $\beta_i = \frac{11}{9}\text{MMS}_i^n$, Huang and Lu [2019] prove that Algorithm 5.3 computes an $\frac{11}{9}\text{MMS}^n$ allocation. However, computing these threshold values is NP-Hard. They further prove that using a PTAS for MMS, it is possible to compute an $\frac{5}{4}\text{MMS}^n$ allocation in polynomial time.

For intuition, we leverage this approximation technique to compute both 2MMS^n and $\frac{3}{2}\text{MMS}^n$ with simple analysis. Suppose that valuations are scaled so that $\min(v_i(b_1), \frac{v_i(M)}{n}) = -1$. Thus we know that $\text{MMS}_i \leq -1$. If we select $\beta_i = -2$ for each agent then run Algorithm 5.3, the resulting allocation is a 2-approximation for MMS.

Proposition 3. *If valuations are scaled so that $\min(v_i(b_1), \frac{v_i(M)}{n}) = -1$ for all $i \in N$, then Algorithm 5.3 with $\beta_i = -2$ for all $i \in N$ returns an allocation where $v_i(A_i) \geq 2MMS_i$ for all $i \in N$.*

Proof. Observe that all agents receive a bundle under Algorithm 5.3 as any agent may always take $\emptyset$. By Lemma 16, $v_i(b) \geq \text{MMS}_i^n$ and $\frac{v_i(M)}{n} \geq \text{MMS}_i^n$ by Lemma 14. Thus the scaling guarantees that $\text{MMS}_i^n \leq -1$ for all agents, and since each agent is allocated a bundle worth more than -2, they receive a bundle worth more than 2MMS_i^n.

Algorithm 5.3: Bag-filling for Chores

Input : An ordered chores instance $\mathcal{I} = \langle N, M, V \rangle$, threshold values of agents $(\beta_1, \ldots, \beta_n)$
Output: An allocation A

Let $M' = M$, $N' = N$, and $A = (\emptyset, \ldots, \emptyset)$;
for $k = 1$ **to** n **do** `// Loop to generate bundles`
  $B = \emptyset$;
  Let $b_1, b_2, \ldots, b_{m'}$ be the order of chores in M';
  for $j = 1$ **to** $|M'|$ **do** `// From the largest chore to the smallest`
    if $\exists i \in N', v_i(B \cup \{b_j\}) \leq \beta_i$ **then**
      $B = B \cup \{b_j\}$;
  Let $i \in N'$ be an agent such that $v_i(B) \leq \beta_i$;
  $A_i = B$; $M' = M' \setminus B$; and $N' = N' \setminus i$;
return A;

Consider the last agent, say agent i, to receive a bundle. Since agent i could not add any remaining chore to any allocated bundle, we see that for each $j \neq i$, $v_i(b) + v_i(A_j) \leq -2$. However, since $v_i(b) \geq -1$, this implies that $v_i(A_j) \leq -1$. Since this holds for all $n-1$ other agents, the total value remaining is at least $v_i(M) - (n-1)(-1) = n\frac{v_i(M)}{n} + (n-1) \geq n(-1) + (n-1) = -1 > 2\text{MMS}_i^n$. Thus agent i can take all remaining chores and still be satisfied. □

Like in the case of goods, a very simple modification to this algorithm guarantees a $\frac{3}{2}$-approximation of MMS with very simple analysis. By Lemma 16, we know that there are at most n chores with value at most $\frac{\text{MMS}_i^n}{2}$. If we initialize bundles so that chores b_1 to b_n all accounted for before bag-filling starts, each chore added during bag filling has value at least $\frac{\text{MMS}_i^n}{2}$. Using this idea, we can improve the simple approximation to $\frac{3}{2}\text{MMS}^n$.

Proposition 4. *Let $I = \langle N, M, V \rangle$ be an ordered chores instance with valuations scaled so that for all $i \in N$,*

$$\min(v_i(b_1), v_i(\{b_n, b_{n+1}\}), \frac{v_i(M)}{n}) = -1.$$

If we initialize bundles so that for all $j \in [n]$, $b_j \in A_j$ and select $\beta_i = -\frac{3}{2}$ for all $i \in N$, then Algorithm 5.3 computes an $\frac{3}{2}MMS^n$ allocation.

Proof. Since $v_i(\{b_n, b_{n+1}\}) \geq \min(v_i(b_1), v_i(\{b_n, b_{n+1}\}), \frac{v_i(M)}{n}) = -1$, we have that $v_i(\{b_n, b_{n+1}\}) \geq -1$. Since the instance is ordered, this implies $v_i(b_{n+1}) \geq -\frac{1}{2}$.

Also, by Lemma 14 and Lemma 16, $\text{MMS}_i^n \leq \min(v_i(b_1), v_i(\{b_n, b_{n+1}\}), \frac{v_i(M)}{n})$. Thus, $\text{MMS}_i^n \leq -1$. Since agents are only allocated bundles worth more than $-\frac{3}{2}$, we only need to show that all items are allocated.

Consider the last agent to receive a bundle, say agent i. Since agent i could not add any remaining item b to any item allocated to other agents, $v_i(A_j) + v_i(b) \leq -\frac{3}{2}$. However, since each remaining chore (other than b_n) has value greater than $-\frac{1}{2}$, we see that $v_i(b) > -\frac{1}{2}$. This implies that for all $j \neq i$, $v_i(A_j) \leq -1$. Since $n-1$ other agents have been allocated bundles, the remaining value is at least $v_i(M) - (n-1)(-1) \geq -n + (n-1) = -1$. Thus agent i can take all remaining chores. □

5.4 Computing $\text{MMS}^{\gamma n}$

In the previous section, we showed how to compute a $\frac{3}{2}\text{MMS}^n$ allocation using simple techniques. With a more complicated analysis, Huang and Lu [2019] proved that Algorithm 5.3 computes a $\frac{5}{4}\text{MMS}^n$ allocation in polynomial time, and with access to an MMS oracle it can compute an $\frac{11}{9}\text{MMS}^n$ allocation. While our analysis does not push the boundaries of maximin share approximations for chores, it introduces the techniques needed to prove a different MMS approximation. In the context of goods, we showed that $\text{MMS}^{\lceil \frac{3}{2}n \rceil}$ exists. Using the techniques from the previous section, we prove that the corresponding approximation exists for chores. That is, each agent can be guaranteed their $\text{MMS}^{\lfloor \frac{2}{3}n \rfloor}$ value.

We first illustrate the nuanced differences between βMMS and MMS^γ. Lemma 13 shows that if q is an integer then $q\text{MMS}^{qk} \leq \text{MMS}^k$. Thus if $\frac{n}{2}$ is an integer, $2\text{MMS}^{2(\frac{n}{2})} \leq \text{MMS}^{\frac{n}{2}}$. However, it is possible that $\text{MMS}^{\frac{n}{2}} > 2\text{MMS}^n$. In this case, we cannot simply guarantee that each agent receives a bundle worth at least 2MMS^n. When considering $\text{MMS}^{\gamma n}$, it is possible that $\text{MMS}^{\gamma n} = \text{MMS}^n$. Consider the following instance: $n+1$ chores all of value -1. For $\gamma \in [\frac{1}{2} + \frac{1}{2n}, 1]$, $\text{MMS}^{\lfloor \gamma n \rfloor} = -2$. Thus relaxing γ does not change the maximin share value.

We observe that $\beta\text{MMS}^n \not\Rightarrow \text{MMS}^{\frac{1}{\beta}n}$. However, we observe that like goods, MMS^k is monotonic in k for chores.

Proposition 5. *If $a \leq b$ then $\text{MMS}^a \leq \text{MMS}^b$.*

Proof. Consider an MMS partition $A = (A_1, \dots A_a)$ for MMS^a. We extend A into a b-partition by adding $b-a$ empty bundles so that $A' = (A_1, \dots, A_a, \emptyset, \dots, \emptyset)$. Observe that the minimum bundle of this partition has value at least MMS^a and thus $\text{MMS}^b = \max_{B=(B_1,\dots,B_b)\in\Pi_b(M)} \min_{j\in[b]} v_i(B_j) \geq \min_{j\in[b]} v_i(A'_j) = \text{MMS}^a$. □

We observe that if $a < b$ and $\text{MMS}^a = \text{MMS}^b$ then the bound $\text{MMS}^b \leq \frac{v_i(M)}{b}$ is not tight as $\text{MMS}^b = \text{MMS}^a \leq \frac{v_i(M)}{a} < \frac{v_i(M)}{b}$. While the possibility of $\text{MMS}^{\gamma n} = \text{MMS}^n$ implies that we may not trivially adapt the βMMS^n algorithm, this looseness gives hope that for another algorithm for $\text{MMS}^{\gamma n}$. In the following theorem, we prove that Algorithm 5.3 can be adapted to compute $\text{MMS}^{\lfloor \frac{2}{3}n \rfloor}$.

Theorem 6. *For additive chores instances, $\text{MMS}^{\lfloor \frac{2}{3}n \rfloor}$ exists and can be computed in polynomial time.*

Proof. Let $q = \lfloor \frac{2}{3}n \rfloor$. We initialize bundles A_1 to A_q with chores b_1 to b_q respectively. Then we pair chores b_{q+1} to b_{2q} to initialize $\lceil \frac{q}{2} \rceil$ bundles (A_{q+1} to $A_{q+\lceil \frac{q}{2} \rceil}$) with at most 2 chores each. It is important to note that $q + \lceil \frac{q}{2} \rceil = \lfloor \frac{2}{3}n \rfloor + \lceil \frac{\lfloor \frac{2}{3}n \rfloor}{2} \rceil \leq \lfloor \frac{2}{3}n \rfloor + \lceil \frac{\frac{2}{3}n}{2} \rceil = \lfloor \frac{2}{3}n \rfloor + \lceil \frac{n}{3} \rceil = n$. Thus we have initialized at most n bundles. By Lemma 16, we know that $v_i(b_1) \geq \text{MMS}_i^q$ and that $v_i(b_q, b_{q+1}) \geq \text{MMS}_i^q$ for all $i \in N$. Since the instance is ordered all initial bundles have value at least MMS^q to all agents. We observe that this is true regardless of how the instance is scaled.

We now scale valuations so that $\frac{v_i(M)}{n} = -1$. For each agent, we select

$$\beta_i = \min\left(v_i(b_1), v_i(\{b_q, b_{q+1}\}), v_i(\{b_{2q-1}, b_{2q}, b_{2q+1}\}), \frac{v_i(M)}{q}\right).$$

By Lemma 16 and Lemma 14, we see that $\text{MMS}_i^q \leq \beta_i$ and that $\beta_i \leq \frac{v_i(M)}{q} = \frac{v_i(M)}{\lfloor \frac{2}{3}n \rfloor} \leq \frac{v_i(M)}{\frac{2}{3}n} = -\frac{3}{2}$.

If we run Algorithm 5.3 using the above initialization, scaling, and β values, we claim that the resulting allocation satisfies MMS^q. It is easy to see that each allocated bundle is worth more than β_i to the agent who received it. Thus $v_i(A_i) \geq \beta_i \geq \text{MMS}_i^q$. Furthermore, since every bundle is initialized so that $v_i(A_j) \geq$

MMS_i^q, all agents may accept bundles as they were initialized. Thus each agent will receive a bundle. All that we need to show now is that all items are allocated when the algorithm terminates.

Consider the last agent to receive a bundle, say agent i. Since agent i was not able to add any remaining chore to any bundle allocated to another agent, we have $v_i(A_j) + v_i(b) \leq \beta_i$ for each remaining chore b. Since the instance is ordered and $v_i(\{b_{2q-1}, b_{2q}, b_{2q+1}\} \geq \beta_i$, we see that $v_i(b_{2q+1}) \geq \frac{\beta_i}{3}$. Combining this with the fact that all remaining chores (other than $b_{2q-1} and b_{2q}$) have value at least $v_i(b_{2q+1})$, we have that each allocated bundle must have value at most $v_i(A_j) \leq \frac{2}{3}\beta_i$. However, because $\beta_i \leq -\frac{3}{2}$, this implies that $v_i(A_j) \leq \frac{2}{3}(-\frac{3}{2}) = -1$. Since this holds for each $j \neq i$, the total amount of remaining value is at most $v_i(M) - (n-1)(-1) \geq -1$. Thus agent i may take all remaining chores and still satisfy $v_i(A_i) \geq -1 \geq \text{MMS}_i^q$. □

5.5 A Comparison with Goods

Maximin share approximations for chores are typically much easier to compute than their counterpart for goods. Perhaps the most important difference between goods and chores comes from the notion of reductions. For goods, we need to have an approximation of the maximin share values before we may commit to a reduction. For chores, the results for reductions hold whether or not we know the maximin share value. For example, we know that if there is a good worth more than the maximin share to some agent, then we may reduce our instance by allocating that good to the interested agent. For chores, we know that there is no chore worth less than the maximin share. If exact approximation is not necessary, we may tolerate an estimate of the maximin shares for goods when we perform this reduction. However, as in the case of Remark 1, it is often the case that even a slight error in the maximin values may prevent algorithms from allocating appropriately. In this situation, we may prove the existence of certain approximations with non-efficient algorithms.

Another important distinction between goods and chores is the notion of desirability. When one agent takes a good with high magnitude, this creates a natural envy from other agents. However, in the case of chores, taking higher magnitude chores only makes other agents less envious of you. As goods become less divisible, agents compete more for the goods; however, as chores become less divisible, agents compete to not take them.

6 Conclusions and Future Directions

6.1 Conclusions

In the traditional MMS approximation setting, it is known that a $(1, \frac{3}{4})$-MMS allocation can always be computed efficiently. We have shown that maximizing this approximation may yield allocations where almost every agent is unsatisfied. Consequently, we proposed a novel fairness notion based on the population of agents. We showed that for small values of n, sacrificing a single agent can increase the fraction guaranteed to other agents, and along the way, we proved that MMS^{n+1} can be computed for 3 agents. We proved that for every goods instance, there always exists an allocation which gives at least $\frac{2}{3}$ of the agents their MMS guarantee, and for $n < 9$, such an allocation can be computed in polynomial time. In practice, we empirically demonstrated that our algorithm for $n < 9$ outperforms its guarantee and computes $(\frac{2}{3}, 1)$-MMS for all but pathological counterexamples. Using the $(\frac{2}{3}, 1)$-MMS existence result, we proved that $\text{MMS}^{\lceil \frac{3}{2} n \rceil}$ exists pushing the previous bound of 1-out-of-$2n-2$ MMS.

In the context of chores, we discussed various approximations of maximin share. We extended the proof that there exists instances for chores where MMS does not exist to any n. We explored the standard lemmas from goods and translated them as closely as possible to chores. We demonstrated simple algorithms for computing $(1, \frac{3}{2})$-MMS for chores using these new lemmas. The ideas of this algorithm were extended into a $\text{MMS}^{\lfloor \frac{2}{3} n \rfloor}$ algorithm for chores.

6.2 Future Directions

Our work provides an important starting point in population based approximations of fairness. In this work, we showed that $\frac{2}{3}$ approximations are achievable for both goods and chores; however, the techniques used to achieve these approximations do not immediately extend into stronger approximations. Therefore, the most natural open question is for what values of α does $(\alpha, 1)$-MMS exist? Similarly, what is the smallest value n' such that $\text{MMS}^{n'}$ exists? In a different setting, where items can be over-allocated, Budish [2011] proved that MMS^{n+1} exists. However, when only feasible allocations are considered, tight upper and lower bounds of n' remain an open problem.

Another important intermediate problem is whether stronger (α, β)-MMS interpolations can be achieved. For example, one might expect that by using the reductions $\{g_1\}, \{g_q, g_{q+1}\}$, we can bag-fill so that each bundle has no more value than $\frac{3}{2} v_i \beta$. With items scaled so that $v_i(M) = n$, we can then aim for $\alpha n \approx \frac{n}{\frac{3}{2}\beta}$ bundles (simplifying gives $\alpha\beta \approx \frac{2}{3}$). This intuition leads to algorithms for $(\frac{2}{3}, 1)$-MMS and $(1, \frac{2}{3})$-MMS; however, it also hints towards interpolated values such as $(\sqrt{\frac{2}{3}}, \sqrt{\frac{2}{3}})$-MMS and $(\frac{4}{5}, \frac{5}{6})$-MMS.

In our research, we offer no guarantee to agents not selected as part of the $\lfloor \alpha n \rfloor$ agents. However, in our experiments, we often saw that after $(\frac{2}{3}, 1)$-MMS was satisfied, there was significant value remaining. While we continued to try to optimize α, another alternative is to maximize β for the remaining agents. In either case, we wish to waste as little value as possible. This leads us to an important question regarding economic efficiency. We know that whenever an (α, β)-MMS allocation exists, so too must an (α, β)-MMS + PO allocation. However, at this point, we have not discussed finding such an allocation. Direct Pareto improvements are NP-Hard to find, so new algorithms which guarantee PO allocations must be considered.

While the notion of (α, β)-MMS doesn't naturally extend to chores, population based approximations did extend to that setting. Another important setting is when we combine goods and chores to form mixed allocations. While Kulkarni et al. [2020] demonstrated that many MMS problems become NP-Hard in

this domain, we first ask which, if any, population based approximations of MMS are meaningful in mixed domains. We also observe that our notions of reductions for both goods and chores settings relied upon the fact that all valuations had the same sign. For example, adding chores to an instance means we can no longer redistribute items when forming reductions for goods. Thus we ask if any of the standard lemmas will extend to mixed settings.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1850076.

References

Elad Aigner-Horev and Erel Segal-Halevi. Envy-free matchings in bipartite graphs and their applications to fair division. *arXiv preprint arXiv:1901.09527*, 2019.

Georgios Amanatidis, Evangelos Markakis, Afshin Nikzad, and Amin Saberi. Approximation algorithms for computing maximin share allocations. *ACM Transactions on Algorithms (TALG)*, 13(4):52, 2017.

Haris Aziz, Gerhard Rauchecker, Guido Schryen, and Toby Walsh. Algorithms for max-min share fair allocation of indivisible chores. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 335–341, 2017.

Haris Aziz, Peter Biro, Jérôme Lang, Julien Lesca, and Jérôme Monnot. Efficient reallocation under additive and responsive preferences. *Theoretical Computer Science*, 790:1–15, 2019.

Moshe Babaioff, Noam Nisan, and Inbal Talgam-Cohen. Fair allocation through competitive equilibrium from generic incomes. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 180–180, 2019.

Siddharth Barman and Sanath Kumar Krishna Murthy. Approximation algorithms for maximin fair division. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 647–664, 2017.

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, Mar 2016. ISSN 1573-7454. doi: 10.1007/s10458-015-9287-3. URL https://doi.org/10.1007/s10458-015-9287-3.

Eric Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011.

Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 305–322. ACM, 2016.

Bhaskar Ray Chaudhury, Jugal Garg, and Kurt Mehlhorn. EFX exists for three agents. In *Proceedings of the 21st ACM Conference on Economics and Computation*, EC '20, page 1–19, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379755. doi: 10.1145/3391403.3399511. URL https://doi.org/10.1145/3391403.3399511.

John P Dickerson, Jonathan Goldman, Jeremy Karp, Ariel D Procaccia, and Tuomas Sandholm. The computational rise and fall of fairness. In *Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*, pages 1405–1411. Citeseer, 2014.

Lester E Dubins and Edwin H Spanier. How to cut a cake fairly. *The American Mathematical Monthly*, 68 (1P1):1–17, 1961.

Francis Edward Su. Rental harmony: Sperner's lemma in fair division. *The American mathematical monthly*, 106(10):930–942, 1999.

Duncan K. Foley. Resource allocation and the public sector. *Yale Economic Essays*, 7:45–98, 1967.

Jugal Garg and Setareh Taki. An improved approximation algorithm for maximin shares. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 379–380, 2020.

Jugal Garg, Peter McGlaughlin, and Setareh Taki. Approximating maximin share allocations. In *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

Mohammad Ghodsi, MohammadTaghi HajiAghayi, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. Fair allocation of indivisible goods: Improvements and generalizations. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 539–556. ACM, 2018.

Jonathan R Goldman and Ariel D Procaccia. Spliddit: unleashing fair division algorithms. *SIGecom Exchanges*, 13(2):41–46, 2014.

Tobias Heinen, Nhan-Tam Nguyen, Trung Thanh Nguyen, and Jörg Rothe. Approximation and complexity of the optimization and existence problems for maximin share, proportional share, and minimax share allocation of indivisible goods. *Autonomous Agents and Multi-Agent Systems*, 32(6):741–778, 2018.

Xin Huang and Pinyan Lu. An algorithmic framework for approximating maximin share allocation of chores. *arXiv preprint arXiv:1907.04505*, 2019.

Rucha Kulkarni, Ruta Mehta, and Setareh Taki. Approximating maximin shares with mixed manna. *arXiv preprint arXiv:2007.09133*, 2020.

David Kurokawa, Ariel D Procaccia, and Junxing Wang. When can the maximin share guarantee be guaranteed? In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 523–529. AAAI Press, 2016.

David Kurokawa, Ariel D Procaccia, and Junxing Wang. Fair enough: Guaranteeing approximate maximin shares. *Journal of the ACM (JACM)*, 65(2):8, 2018.

Richard J Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 125–131. ACM, 2004.

Nhan-Tam Nguyen, Trung Thanh Nguyen, and Jörg Rothe. Approximate solutions to max-min fair and proportionally fair allocations of indivisible goods. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 262–271. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

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. ACM, 2014.

Andrew Searns and Hadi Hosseini. Fairness does not imply satisfaction (student abstract). *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(10):13911–13912, Apr. 2020. doi: 10.1609/aaai.v34i10.7228. URL https://ojs.aaai.org/index.php/AAAI/article/view/7228.

Erel Segal-Halevi. The maximin share dominance relation. *arXiv preprint arXiv:1912.08763*, 2019.

Hugo Steinhaus. The problem of fair division. *Econometrica: Journal of the Econometric Society*, pages 315–319, 1949.

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

A Polynomial Time Approximation Schemes (PTAS)

A polynomial time approximation scheme (PTAS) is an approximation algorithm that offers a tradeoff between running time and accuracy. Typically in order to have an error tolerance of ϵ, a PTAS will incur an additional $\log(\frac{1}{\epsilon})$ factor in the running time of the algorithm.

In the context of MMS, multiple PTAS algorithms exist for various tasks. Woeginger [1997] provided a PTAS algorithm for computing $(1-\epsilon)\text{MMS}_i^n$. This often leads to algorithms for $(1, (1-\epsilon)\beta)$-MMS approximations of MMS. For example, Garg and Taki [2020], Ghodsi et al. [2018] provide algorithms for computing $(1, \frac{3}{4}-\epsilon)$-MMS which run in time $\text{poly}(m, n, \log(\frac{1}{\epsilon}))$. Heinen et al. [2018] demonstrate a PTAS algorithm for approximating optimal-MMS: $(1, (1-\epsilon)\lambda^*)$-MMS. Often, this algorithm is sufficient for finding MMS allocations (when $\lambda^* > \frac{v_i(M)}{n}$). However, this algorithm cannot be used to efficiently determine whether an MMS allocation exists, and when MMS doesn't exist, this algorithm may perform arbitrarily poorly with respect to α (See Theorem 1).

B Material Omitted from Section 4.2

This section includes all the detailed discussions on constructing tensors S and T, the relevant lemmas, and the missing proofs.

We begin with the generalized version of Theorem 1. We observe that when $d = \lfloor \frac{n}{2} \rfloor$ the following theorem becomes Theorem 1.

Theorem 7. *For any $n \geq 4$ and any $d \leq \lfloor \frac{n}{2} \rfloor$, there exists an instance with $O(dn)$ goods where any optimal-MMS allocation guarantees at most $\lceil \frac{n}{d} \rceil + 1$ agents their MMS value.*

Our construction is broken into three components: a tensor S, a tensor T, and d groups of agents. The entries of tensor S are chosen such that equi-partitions of the entries correspond to a set of slices of S. An equi-partition is a partition of the entries of a tensor where each set of the partition has the same sum. Here, all equi-partitions refer to n-partitions. Tensor T provides perturbation so that the only equi-partitions of $S + T$ are aligned slices.

All tensors in our construction are order d with dimensions $(n \times n \times \ldots \times n)$.[11] We say that tensor S is indexed by a d-tuple $(x_1, \ldots, x_d)$ corresponding to entry $S[x_1, \ldots, x_d]$ or $S_{x_1,\ldots,x_d}$ for short. The $(d-1)$-order slices of tensor S along dimension j are given by $S_j(1)$ to $S_j(n)$ where $S_j(i) = \{S_{x_1 \ldots x_d} \mid x_j = i\}$. We also define $S_{i;x_j=k}$ to be the entry where all indices are i except index $x_j = k$. For example, $S_{i;x_1=n} = S_{nii\ldots i}$. Tensor addition is entry-wise $((S+T)_{x_1,x_2,\ldots,x_d} = S_{x_1,x_2,\ldots,x_d} + T_{x_1,x_2,\ldots,x_d})$. Unless otherwise specified, the entries of a tensor have value 0.

Figure 9a depicts the indexing of an order 3 tensor with dimensions $(6 \times 6 \times 6)$. Figure 9b illustrates the distinct sets of slices of such a tensor. Lastly, Figure 9c depicts the non-zero entries of tensors S and T in our construction when $n = 6$ and $d = 3$.

B.1 Construction of Tensor S

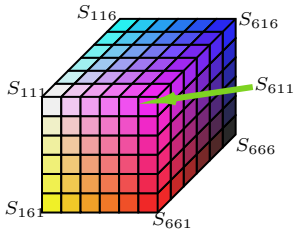
Let S be an order d tensor whose non-zero entries are as follows:

- For $1 \leq i < n$ and $j \in [d]$, $S_{ii\ldots i} = \frac{d^{n-i}-1}{d^{n-i}}$
- For $1 \leq i < n$ and $j \in [d]$, $S_{i;x_j=n} = \frac{1}{d-1} \cdot \frac{1}{d^{n-i}}$
- $S_{nn\ldots n} = 1 - \sum_{i=1}^{n-1} \frac{1}{d-1} \cdot \frac{1}{d^{n-i}}$

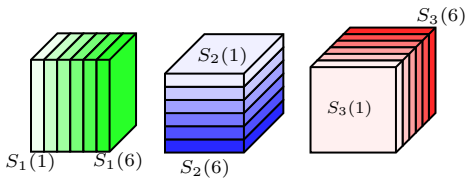
Lemma 18. *Each $(d-1)$-order slice $S_i(j)$ has sum 1.*

Proof. Consider the non-zero entries within slice $S_j(i)$. First when $i \neq n$, the only non-zero entries of $S_j(i)$

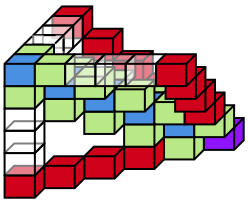
[11] When referring to a tensor, the "order" is the number of dimensions that the tensor is embedded in. The "dimensions" of a tensor, however, refers to the number of entries along each axis, similar to the dimensions of a matrix.



(9a) The indexing of tensor S.



(9b) The slices of tensor S.



(9c) The non-zero entries of S and T.

Figure 9: The shape and slices of order 3 tensors with dimensions $(6 \times 6 \times 6)$.

are $S_{ii\ldots i}$ and $S_{i;x_k=n}$ for each $k \neq j$. Thus:

$$\begin{aligned}
\sum_{x \in S_j(i)} x &= S_{ii\ldots i} + \sum_{k \neq j}^{d} S_{i;x_k=n} \\
&= \frac{d^{n-i}-1}{d^{n-i}} + \sum_{k \neq j}^{d} \frac{1}{d-1} \cdot \frac{1}{d^{n-i}} \\
&= \frac{d^{n-i}-1}{d^{n-i}} + (d-1)\frac{1}{d-1} \cdot \frac{1}{d^{n-i}} \\
&= \frac{d^{n-i}-1}{d^{n-i}} + \frac{1}{d^{n-i}} \\
&= 1
\end{aligned}$$

When $i = n$, the non-zero entries of slice $S_j(i)$ are $S_{nn\ldots n}$ and $S_{k;x_j=n}$ for each $k \neq n$. Thus:

$$\begin{aligned}
\sum_{x \in S_j(n)} x &= S_{nn\ldots n} + \sum_{k \neq n}^{n} S_{k;x_j=n} \\
&= 1 - \sum_{i=1}^{n-1} \frac{1}{d-1} \cdot \frac{1}{d^{n-i}} + \sum_{k=1}^{n-1} \frac{1}{d-1} \cdot \frac{1}{d^{n-k}} \\
&= 1
\end{aligned}$$

□

The combinatorial structure of S has a few other desirable properties. First observe that the entries on the main diagonal $(S_{ii\ldots i})$ all have value greater than $\frac{1}{2}$. When considering equi-partitions of the entries of S, these entries must all be included in separate sets. Furthermore, the entries off the main diagonal increase when approaching $S_{nn\ldots n}$ in such a way that all d of the entries with value $\frac{1}{d-1} \cdot \frac{1}{d^{n-i}}$ are required to account for a single entry with value $\frac{1}{d-1} \cdot \frac{1}{d^{n-(i+1)}}$.

Lemma 19. *The sets of slices $\{S_{j_1}(1), S_{j_2}(2), \ldots, S_{j_n}(n)\}$ are the only equi-partitions equi-partition of the non-zero entries of S.*

Proof. Lemma 18 implies that a set of slices of the form $\{S_{j_1}(1), S_{j_2}(2), \ldots, S_{j_n}(n)\}$ forms an equi-partition. In order to show that these equi-partitions are unique, consider the set which contains $S_{11\ldots1}$. Since $1 - S_{11\ldots1} = \frac{1}{d^{n-1}} < \frac{1}{d-1} \cdot \frac{1}{d^{n-2}}$. Thus the only entries which can be included with $S_{11\ldots1}$ are of the form

$\frac{1}{d-1} \cdot \frac{1}{d^{n-1}}$. Importantly, $d-1$ of these entries uniquely identify a $(d-1)$-order slice $S_{j_1}(1)$ where $S_{1;x_{j_1}=n}$ is the missing entry with value $\frac{1}{d-1} \cdot \frac{1}{d^{n-1}}$.

By induction, with a similar argument at each step, each successive entry on the main diagonal must be included in a set which corresponds exactly to a slice $S_{j_i}(i)$. □

B.2 Construction of Tensor T

In Lemma 19, the slices of S are not necessarily aligned slices. For example, in the case of three dimensions, the first slice might be frontal while the second one might be vertical. Since all slices have the same sum, there is no reason an agent wouldn't compute MMS by taking slices at random. In order to rectify this, we construct tensor T so that equi-partitions of $S+T$ correspond only to aligned slices.

Let tensor T be an order d tensor with non-zero values as follows:

- For $1 \leq i < n-1$, $j \in [d]$, $T_{i;x_j=n} = -r_{ij}$
- For $1 \leq i < n-1$, $j \in [d]$, $T_{i;x_j=i+1} = u_{ij}$
- For $j \in [d]$, $T_{n-1;x_j=n} = x_j$
- $T_{nn\ldots n} = z_1$

Where

- $r_{ij} = \epsilon^{d(n-1)-di-j+1}$
- $u_{1j} = r_{1j}$
- $u_{ij} = \frac{1}{d-1}\left(\sum_{k\neq j}^{d} -u_{i-1,k}\right) + \frac{d-2}{d-1}u_{i-1,j} + r_{ij} \approx r_{ij}$
- $x_j = u_{n-1,j} - r_{n-1,j}$
- $z_j = \left(\sum_{i=1}^{n-2} r_{ij}\right) - x_j$

It is important to observe that the entries of T are all proportional to powers of a very small ϵ term. While some of these values are negative, the negative entries only occur in indices where S has positive value. Thus if ϵ is small enough, $S+T$ has only non-negative values. Another important observation is that the slices of T all have sum 0.

Lemma 20. *Each $(d-1)$-order slice $T_j(i)$ has sum 0.*

Proof. We first address the case of $i=1$. Slice $T_j(1)$ contains the non-zero entries $T_{1;x_k=n}$ for each $k \neq j$ and $T_{1;x_k=2}$ for each $k \neq j$. For each $k \neq j$, the entry $T_{1;x_k=n} = -r_{1j}$ and $T_{1;x_k=2} = u_{1j}$. Since $u_{1j} = -r_{1j}$ the sum of these two entries is 0. Adding over all $k \neq j$ yields that the first slice (independent of j) has sum 0.

Now consider slice $T_j(i)$ for $1 < i < n-1$. The non-zero entries of T in this slice are $T_{i;x_k=n}$ and $T_{i;x_k=i+1}$

for $k \neq j$ and $T_{i-1;x_j=i}$. Adding these entries together yields:

$$\begin{aligned}
&\sum_{x \in T_j(i)} x \\
&= \sum_{k \neq j}^{d} T_{i;x_k=i+1} + \sum_{k \neq j}^{d} T_{i;x_k=n} + T_{i-1;x_j=i} \\
&= \sum_{k \neq j}^{d} (u_{ik}) + \sum_{k \neq j}^{d} (-r_{ik}) + u_{i-1,j} \\
&= \sum_{k \neq j}^{d} \left(\frac{1}{d-1} \left(\sum_{l \neq k}^{d} -u_{i-1,l} \right) + \frac{d-2}{d-1} u_{i-1,k} + r_{ik} \right) \\
&\quad + \sum_{k \neq j}^{d} (-r_{ik}) + u_{i-1,j} \\
&= \sum_{k \neq j}^{d} \left(\frac{1}{d-1} \left(\sum_{l \neq k}^{d} -u_{i-1,l} \right) + \frac{d-2}{d-1} u_{i-1,k} \right) + u_{i-1,j} \\
&= \frac{1}{d-1} \left(\sum_{k \neq j}^{d} \sum_{l \neq k}^{d} (-u_{i-1,l}) + (d-2) \sum_{k \neq j}^{d} (u_{i-1,k}) \right) + u_{i-1,j} \\
&= \frac{1}{d-1} \left((d-1)(-u_{i-1,j}) \right) + u_{i-1,j} = 0
\end{aligned}$$

The last step follows because each of the $(d-1)$ summations (for each $k \neq j$) adds a single $-u_{i-1,j}$ term. For each $k \neq j$, $(d-2)$ of the summations (all but $l = k$ of the $(d-1)$ inner summations) add a $-u_{i-1,k}$ term.

We next address the case where $i = n-1$. In this case, we observe that $i+1 = n$ so the only entries in

this slice are $T_{n-1;x_k=n}$ for $k \neq j$ and $T_{n-2;x_j=n-1}$. Adding these entries together yields:

$$\begin{aligned}
&\sum_{x \in bi T_j(n-1)} x \\
&= \sum_{k \neq j}^{d} T_{n-1;x_k=n} + T_{n-1;x_j=n-1} \\
&= \sum_{k \neq j}^{d} (x_k) + u_{n-2,j} \\
&= \sum_{k \neq j}^{d} (u_{n-1,k} - r_{n-1,k}) + u_{n-2,j} \\
&= \sum_{k \neq j}^{d} \left(\left(\frac{1}{d-1} \left(\sum_{l \neq k}^{d} -u_{n-2,l} \right) \right. \right. \\
&\qquad \left. \left. + \frac{d-2}{d-1} u_{i-2,k} + r_{n-1,k} \right) - r_{n-1,k} \right) + u_{n-2,j} \\
&= \sum_{k \neq j}^{d} \left(\frac{1}{d-1} \left(\sum_{l \neq k}^{d} -u_{n-2,l} \right) + \frac{d-2}{d-1} u_{i-2,k} + \right) + u_{n-2,j} \\
&= \frac{1}{d-1} \left(\sum_{k \neq j}^{d} \sum_{l \neq k}^{d} (-u_{n-2,l}) + (d-2) \sum_{k \neq j}^{d} (u_{i-2,k}) + \right) + u_{n-2,j} \\
&= \frac{1}{d-1} ((d-1)(-u_{n-2,j})) + u_{n-2,j} = 0
\end{aligned}$$

We last address the case where $i = n$. In this case, the non-zero entries in this slice are $T_{i;x_j=n}$ for each $i \in [n-2]$, $T_{n-1;x_j=n}$, and $T_{nn\ldots n}$. Adding these entries together yields:

$$\begin{aligned}
\sum_{x \in T_j(n)} x &= \sum_{k=1}^{n-2} T_{k;x_j=n} + T_{n-1;x_j=n} + T_{nn\ldots n} \\
&= \sum_{k=1}^{n-2} (-r_{kj}) + x_j + z_1
\end{aligned}$$

If $j = 1$ then the term $\sum_{k=1}^{n-2} (-r_{kj}) + x_j = -z_1$ so this trivially sums to 0. If $j \neq 1$, then we first observe that the sum of all entries of T is 0 by adding the sums of the slices $\{T_1(1), T_1(2), \ldots, T_1(n)\}$. Since $\sum_{i=1}^{n} \sum_{x \in T_j(i)} x = 0$ and $\sum_{k=1}^{n-1} \sum_{x \in T_j(k)} x = 0$, subtracting the two equations gives $\sum_{x \in T_j(n)} = 0$. □

Combining Lemmas 19 and 20 together immediately shows that any set of aligned slices of $(S+T)$ forms an equi-partition of the non-zero entries where each set has sum 1. We now show that these partitions are unique. For intuition, the key point in our construction is that $(d-1)$ of the $T_{i;x_j=i+1}$ terms all lie in the same slice $T_j(i)$. Once j has been chosen for $T_j(1)$ the extra term $T_{1;x_j=2}$ lies in slice $T_j(2)$.

Lemma 21. *The only equi-partitions of the non-zero entries of $(S+T)$ correspond to aligned slices.*

Proof. First observe that if ϵ is sufficiently small then the structure of Lemma 19 still holds. Consider first the set of an equi-partition which contains $(S+T)_{11\ldots1}$. In order for this set to have value 1, it must have $(d-1)$

of the entries which receive $\frac{1}{d-1} \cdot \frac{1}{d^{n-1}}$ value from S. Again, selecting $(d-1)$ of these uniquely characterizes a slice $(S+T)_j(1)$. Observe that these entries have negative perturbations from T corresponding to $-r_{1k}$ for each $k \neq j$. In order for these perturbations to be offset so that the set containing $(S+T)_{11\ldots1}$ has sum 1, the corresponding entries with values u_{1k} for $k \neq j$ must be included in this set. These entries correspond exactly to the slice $(S+T)_j(1)$.

As in the proof of Lemma 19, we observe that the entries on the main diagonal $(S+T)_{ii\ldots i}$ must be in separate sets. Likewise, because ϵ is small, a similar argument enforces that $(d-1)$ of the entries with value $\frac{1}{d-1} \cdot \frac{1}{d^{n-i}}$ must be included in this set. We address the negative perturbations (from T) of these entries by induction. Observe that the smallest perturbations from T occur in the earliest slices ($(1,1) = \arg\min_{i,j} r_{ij}$). Note also that $(u_{1j}, -r_{ij})$ is the only pair of perturbations from T which are excluded from slice $(S+T)_j(1)$. A similar argument for slice $(S+T)_j(1)$ holds for slice $(S+T)_j(2)$: in order for the set containing $(S+T)_{22\ldots2}$ to have value 1, it must also have $(d-1)$ entries with the $-r_{2k}$ perturbations from T. Since $\epsilon^2 \ll \epsilon$, each $-r_{2k}$ entry can only be offset with entries with perturbation u_{2k}. (All but one of the smaller perturbation entries have already been assigned to slice 1.) However, by Lemma 20, we see that adding together all $k \neq l$ entries of the form $-r_{2k}$ and u_{2k} leaves value u_{1l} (since $u_{1l} + \sum_{k\neq l}^{d} u_{2k} + \sum_{k\neq l}^{d} -r_{2k} = 0$). Observe that only a single entry of the form u_{1l} is available: notably u_{1j}. Thus the set which contains $(S+T)_{22\ldots2}$ corresponds to the slice $(S+T)_j(2)$.

By induction on this same argument we see that each set containing the entry $(S+T)_{ii\ldots i}$ for $1 \leq i < n$ must correspond to a slice $(S+T)_j(i)$ for some fixed j (after j is chosen for the first slice). We conclude this proof by observing that the remaining entries all lie in the last slice $(S+T)_j(n)$ and have total sum 1 by Lemmas 19 and 20. □

With the details of S and T, we are now ready to prove the main theorem. We restate it here for convenience.

B.3 Proof of Theorem 7

Theorem 7. *For any $n \geq 4$ and any $d \leq \lfloor \frac{n}{2} \rfloor$, there exists an instance with $O(dn)$ goods where any optimal-MMS allocation guarantees at most $\lceil \frac{n}{d} \rceil + 1$ agents their MMS value.*

Proof. Lemma 21 implies that there are d unique equi-partitions of the entries of $(S+T)$. We group agents arbitrarily into d groups of size at least 2 and at most $\lceil \frac{n}{d} \rceil$ each. We next show how to perturb $(S+T)$ for each group so that each group has a unique equi-partition. Furthermore, we enforce that the only entry with positive perturbation for all agents is the last entry on the main diagonal. Let P^j be an order d tensor with non-zero values as follows:

- For $1 \leq i < n$, $P^j_{i;x_j=n} = -\tilde{\epsilon}$
- $P^j_{nn\ldots n} = (n-1)\tilde{\epsilon}$

The entries where P^j is negative correspond to entries where $(S+T)$ has positive value. Thus $(S+T+P^j)$ has only non-negative entries if $\tilde{\epsilon}$ is sufficiently small. Observe that the slices $\{(S+T+P^j)_j(1), (S+T+P^j)_j(2), \ldots, (S+T+P^j)_j(n)\}$ form an equi-partition of the entries of $(S+T+P^j)$ as P^j has non-zero values only in the last slice $(S+T+P^j)_j(n)$ and the total sum of its non-zero entries is 0. If $\tilde{\epsilon}$ is sufficnetly small (so that the structure of $S+T$ dominates when creating an equi-partition), then none of the other sets of slices of $(S+T+P^j)$ forms an exact equi-partition as every slice except for the last contains exactly one of the $-\tilde{\epsilon}$ perturbations.

We now create an instance $I = \langle N, M, V \rangle$ where M contains one good for each non-zero entry of $(S+T)$. Agents in group j value the goods according to the values of $(S + T + P^j)$. Since every agent is able to form an equi-partition of the entries of their valuation tensor, all agents have $\text{MMS}_i = 1$. By allocating goods according to slices, each agent receives at least $(1 - \tilde{\epsilon})$ of their MMS value. Since $\tilde{\epsilon}$ is very small, the only way all agents can receive at least $1 - \tilde{\epsilon}$ value is if the goods are allocated according to aligned slices (Lemma 21). Under this allocation, at most one group and a single additional agent who receives the last slice (since $(S+T+P^j)_j(n)$ has positive perturbation $(n-1)\tilde{\epsilon}$) receive their MMS value. Lastly we observe that there only $O(dn)$ goods in M. □

Again we note that if $d = \lfloor \frac{n}{2} \rfloor$ in the above theorem, then we see that only 3 agents (4 if n is odd) receive their full MMS in any optimal-MMS allocation.

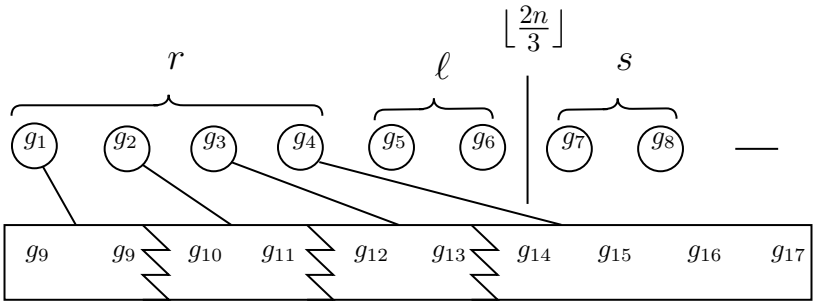


Figure 13: The low-value goods g_9 to g_{17} use a moving-knife procedure to be added to bundles initialized with a single high-value good (g_1 to g_8 each. Thus $h = 8$). The bag-filling algorithm fills r bundles with ℓ left to fill when it runs out of low-value goods. The remaining value is tied up with the surplus s high-value goods.

C Material Omitted from Section 4.5

C.1 Proof of Theorem 3

Theorem 3. *$(\frac{2}{3}, 1)$-MMS exists and can be computed in polynomial time with an oracle for MMS.*

Given the previous lemma, we now show that Algorithm 4.7 can guarantee MMS$\frac{2}{3}1$. We first order and scale the instance so that $\text{MMS}_i = 1$ for all agents. This implies that $v_i(M) \geq n$. Using the $\{g_1\}$ and $\{g_n, g_{n+1}\}$ reduction, we may assume no good has value more than 1, there are at most n high-value goods, and no good has value more than 1. We define a good to be high-value if any agent values it at least $\frac{1}{2}$. A good is low-value otherwise. Let H be the set of h high-value goods.

We begin by letting each agent form preview bundles by running the bag-filling procedure assuming all agents' preferences are identical to their own. Since goods are added to bundles in the same order when agents form preview bundles as in bag-filling, these bundles form the worst case for bag-filling. With this intuition, we first show that each agent is able to form at least $\lfloor \frac{2n}{3} \rfloor - s$ preview bundles worth at least 1 and at most $\frac{3}{2}$.

Lemma 22. *Given a fixed set H, each agent can create at least $\lfloor \frac{2n}{3} \rfloor - s$ preview bundles worth at least 1 and at most $\frac{3}{2}$ using bag-filling.*

Proof. Suppose that agent i's preview bag-filling procedure stopped before $\lfloor \frac{2n}{3} \rfloor$ bundles were filled. We consider cases based on the number of surplus high-value goods $s = \max(h - \lfloor \frac{2n}{3} \rfloor, 0)$ and the number of bundles that were filled r. Define the number of bundles left to be filled as $\ell = \lfloor \frac{2n}{3} \rfloor - r$. Figure 13 depicts an example stopping point of bag-filling with the values of r, ℓ, h, and s respectively.

Since each allocated bundle has value no more than $\frac{3}{2}$, the amount of remaining value is at least $n - \frac{3}{2}r = n - \frac{3}{2}(\lfloor \frac{2n}{3} \rfloor - \ell) = n - \frac{3}{2}\lfloor \frac{2n}{3} \rfloor + \frac{3}{2}\ell \geq \frac{3}{2}\ell$.

1. If $h \leq \lfloor \frac{2n}{3} \rfloor$ then we claim that $\lfloor \frac{2n}{3} \rfloor$ bundles were filled during bag-filling. In this case, the total value from high-value goods is at most $\ell v_i(g_{r+1}) \leq \ell$ (observe that $s = 0$) which leaves at least $\frac{3}{2}\ell - \ell = \frac{1}{2}\ell \geq \frac{1}{2}$ value remaining. If $v_i(g_{r+1}) + \frac{1}{2} < 1$, then the total value from low-value goods is at least $\frac{3}{2}\ell - \frac{1}{2}\ell \geq \ell > 1$ which implies another bundle can still be filled.
2. Suppose that $\ell > s$. We first show that in this case the allocated high-value goods must each be valued at least $\frac{3}{4}$. Suppose for contradiction that $v_i(g_{r+1}) \leq \frac{3}{4}$. Since the instance is ordered and we add high-value goods to bundles in descending order, all remaining high-value goods have value at most $\frac{3}{4}$. Thus the total remaining value from low-value goods is at least $\frac{3}{2}\ell - (\ell + s)v_i(g_{r+1}) \geq \frac{3}{2}\ell - \frac{3}{4}(\ell + s) =$

$\frac{3}{4}\ell - \frac{3}{4}s = \frac{3}{4}(\ell - s) \geq \frac{3}{4}$. The last step follows because ℓ and s are both integers so $\ell > s$ implies $\ell \geq s + 1$. If g_{r+1} together with the remaining low-value goods has value at least 1 (enough to fill a bundle) then we arrive at our contradiction that bag-filling had stopped without allocating these goods. If g_{r+1} with the remaining low-value goods does not fill another bundle then $v_i(g_{r+1}) < \frac{1}{4}$ implying that the total remaining value from low-value goods is at least $\frac{3}{2}\ell - \frac{1}{4}(\ell + s) \geq \frac{3}{2}\ell - \frac{1}{4}(2\ell) \geq \ell \geq 1$. This implies that another bundle could be filled, again contradicting that bag-filling stopped. We may now assume that if $\ell > s$ then $v_i(g_{r+1}) \geq \frac{3}{4}$ for agent i. Since the instance is ordered, this in turn implies that each of the first r high-value goods also had value at least $\frac{3}{4}$.

During bag-filling, if a single low-value good is allocated to a bundle, then that bundle has at most $\frac{1}{2}$ value from low-value goods. However, if multiple low-value goods are added to a bundle, then since low-value goods are added in descending order, the last good has value no more than the second last good. Since the second last good was not enough to fill the bundle, then the total value from the last two low-value goods is no more than twice the value needed to fill the bundle before the last two goods were added. Accordingly, this implies that the total value from low-value goods allocated to a bundle is no more than twice the value of $1 - v_i(g_k)$ where g_k is the initial high-value good in the bundle. Since all bundles allocated during bag-filling contained a high-value good worth at least $\frac{3}{4}$, this implies that the total value of low-value goods allocated to any bundle is at most $\frac{1}{2}$ whether a single low-value good or multiple low-value goods were needed.

Observe that because bags are initialized with high-value goods in descending order, and low-value goods are added to a bundles in descending order, that if ever a single bundle requires multiple low-value goods then each subsequent bundle will also require multiple low-value goods. Likewise, if some bundle requires only a single low-value good, then each bundle allocated before it must also have had only a single low-value good.

We now consider an MMS partition $A = (A_1, \ldots A_n)$ for agent i. Suppose first that there is no bundle in M which contains multiple high-value goods, and suppose without loss of generality that the bundles are sorted such that g_1 is in bundle A_1 and g_h is in A_h. Thus there are at least $n - h$ bundles which contain only low-value goods. These bundles can be separated during bag-filling to fill at least $2(n-h)$ bundles since each bundle filled during bag-filling needed at most $\frac{1}{2}$ value from low-value goods. We also observe that the low-value goods from bundles A_{r+1} to A_h can be paired together to form at least s sets of low-value goods which value at least twice $1 - v_i(g_{r+1})$ each (observe that since $\ell > s$, $\lfloor \frac{s+\ell}{2} \rfloor \geq s$). Thus if at least two low-value goods are needed to fill any bundle after the first $2(n - h)$ bundles, then we can fill an additional s bundles using the value from these low-value bundles. In total we are able to fill at least $2(n - h) + s$ bundles implying that $r \geq 2(n - h) + s$. By definition of r and h this implies that $\lfloor \frac{2n}{3} \rfloor - \ell \geq 2n - 2(\lfloor \frac{2n}{3} \rfloor + s) + s$, implying that $s \geq \ell + 2n - 3\lfloor \frac{2n}{3} \rfloor \geq \ell$ contradicting the assumption that $\ell > s$.

A challenge arises if bundles after the first $2(n - h)$ still require only a single low-value good. It is possible that these goods are valued more than $2(1 - v_i(g_{r+1}))$ and thus prevent us from using our previous argument. Importantly, we may set a threshold that a bundle requires only a single low-value good if the low-value good is worth at least $(1 - v_i(g_{r+1}))$. We observe that any such items had to go into some bundle of the MMS partition A. Consider a low value good g which filled a bundle on its own. If $g \in A_j$ where $j \leq r$ then we are fine because we previously were not counting any of the value from low-value goods within these bundles. If $g \in A_j$ where $r < j \leq h$ then we modify our pairing argument (from the proceeding paragraph) to exclude bundle A_j and pair the remaining bundles.

Suppose that there are k such bundles from A_{r+1} to A_h which contain a good worth more than $(1 - v_i(g_{r+1}))$. Then we may form $k + \lfloor \frac{2s-k}{2} \rfloor \geq k + (s - k) = s$ bundles during bag-filling using the modified pairing argument. Lastly, if $g \in A_j$ for $j > h$, we must be careful to prevent double counting the value from these bundles. If A_j contains two or more low-value goods with value at least $(1 - v_i(g_{r+1})$ then we may safely assume that we fill 2 bundles during bag-filling while still attributing the value from g to A_j. However, if A_j contains less than 2 low-value goods worth at least $(1 - v_i(g_{r+1}))$ then we observe that g is the only such good in A_j. In this case, we may swap a subset of low-value goods worth between $\frac{1}{4}$ and $\frac{1}{2}$ with some low-value good worth at least $1 - v_i(g_{r+1})$ from the bundles A_{r+1} to A_h. Observe that such a subset of goods must exist as there is at least $\frac{1}{2}$ value from low-value goods not worth at least $2(1 - v_i(g_{r+1}))$. If none of these goods are worth more than $\frac{1}{4}$ then a combination of them must be worth between $\frac{1}{4}$ and $\frac{1}{2}$. After performing these swaps we may still assume that each bundle from A_{h+1} to A_n has at least enough value to fill 2 bundles during bag-filling. Any extra low-value goods worth at least $(1 - v_i(g_{r+1}))$ are now accounted for either because they are in bundles A_1 to A_r, because we can use the modified pairing argument, or because they fall in a bundle of A_{h+1} to A_n which contains more than 2 such goods. In total, we now see that agent i is able to fill at least $2(n - h) + s$ bundles during bag-filling. With the same argument as before, this yields the contradiction that $\ell > s$.

We lastly observe that the assumption that A contains no bundle with two high-value goods can be removed as each such bundle causes another bundles of A to have no high-value goods (by pigeonhole principle). In total we are able to fill two more bundles using the low-value goods for each of these additional bundles of A which contain no high-value goods. Simultaneously we may lose at most one bundle from the range A_{r+1} to A_h that we can use for the pairing argument. This net change only increases the number of bundles we may fill during bag-filling. Thus we conclude with the contradiction that $\ell > s$ in all cases which implies that bag-filling never stops when $s > \ell$.

□

We now observe that agent i's preview bundles are a pessimistic preview of bag-filling. Let $B^i = (B^i_1, \ldots, B^i_{\lfloor \frac{2n}{3} \rfloor})$ be the set of preview bundles for agent i, and A_1 to A_r be the bundles allocated by bag-filling (lines 10-12 of Algorithm 4.7).

Lemma 23. *For any agent $i \in N$ who has not yet received a bundle, the set of goods allocated in the first k bundles of bag-filling is a subset of that agent's first k preview bundles. Formally, for each k from 1 to r, $\cup_{j=1}^{k} A_j \subseteq \cup_{j=1}^{k} B^i_j$.*

Proof. Consider for induction the base case of $k = 1$. Since goods are allocated in the same order as they were during preview bundles, each agent would request the first bundle at the same time as they requested their first preview bundle. The agent with the smallest cardinality first bundle will request it before any other agent. Thus for each agent $i \in N$, $A_1 \subseteq B^i_1$.

Suppose for induction that each agent $i \in N$ who has not yet received a bundle believes that $\cup_{j=1}^{k-1} A_j \subseteq \cup_{j=1}^{k-1} B^i_j$. Since the high-value goods are added to successive bundles in descending order, the high value good in B^i_k will be the same high-value good as in A_k, namely g_k. Consider the set of low-value goods in B^i_k. Before bundle A_k is allocated, these goods are necessarily still available as $(M \setminus \cup_{j=1}^{k-1} A_j) \supseteq (M \setminus \cup_{j=1}^{k-1} B^i_j)$. While there may be low-value goods available in $(M \setminus \cup_{j=1}^{k-1} A_j)$ that are not available from $(M \setminus \cup_{j=1}^{k-1} B^i_j)$, we observe that these low-value goods have value at least as much as the low value goods in B^i_k and thus will be added to A_k before any low-value good of B^i_k. Thus either another agent will request A_k before the

last low-value good of B_k^i is added to A_k or agent i will request bundle A_k after the last low-value good of B_k^i has been added to A_k. As a result, either agent i will be allocated a bundle and thus be removed from the set of agents still without a bundle or $\cup_{j=1}^{k} A_j \subseteq \cup_{j=1}^{k} B_j^i$. □

We note that the worst case (which leads to the most competition) in the previous lemma is when all agents have identical preferences. However, in this case, MMS necessarily exists. We now show how to combine Lemma 22 and Lemma 23 to prove that $\text{MMS}\frac{2}{3}1$ exists.

Theorem 3. *$(\frac{2}{3}, 1)$-MMS exists and can be computed in polynomial time with an oracle for MMS.*

Proof. Because of Lemma 3, we only focus on ordered instances. If we run the bag-filling algorithm, Lemma 23 implies that at each step, the set of available goods includes the set of available goods as each agent had when they form preview bundles. Thus Lemma 22 still applies as the arguments in each case still hold.

After bag-filling terminates it must be the case that $s \geq \ell$. Since there exists at least one agent who values g_h at least $\frac{1}{2}$ by definition, we may give that agent the bundle $\{g_{r+1}, g_h\}$. Any agent who does not believe that g_h is worth at least $\frac{1}{2}$ will believe that the allocated bundle has value at most $\frac{3}{2}$. Since $s \geq \ell$, we see that $\lfloor \frac{s+\ell}{2} \rfloor \geq \ell$. Thus, if we can repeat this process at least ℓ times by allocating the first and last high-value goods together, then we will be able to satisfy $\lfloor \frac{2n}{3} \rfloor$ bundles in total. However, suppose that after k steps, no remaining agent believes that $\{g_{r+k}, g_{h-k}\}$ is worth at least 1. In this case, we may truncate the pool of high-value items by redefining some items to be low-value. At this point, the assumption that low-value goods are allocated in descending order will be violated, which may break the analysis of case 2 of Lemma 22 if we resume bag-filling. In order to fix this, we instead perform another iteration of bag-filling, keeping fixed only the bundles allocated by pairing high-value goods. The analysis of Lemma 22 will now hold as any bundle allocated with a single low-value good in this step had at most $\frac{1}{2}$ value from low-value goods to the remaining agents. Because the set of high-value goods strictly decreases in size whenever we repeat this step, the algorithm will eventually terminate with an allocation that satisfies $\text{MMS}\frac{2}{3}1$ on the ordered instance.

We observe that under the assumption that $k_i = \text{MMS}_i$ for each agent, Algorithm 4.7 finds a $\text{MMS}\frac{2}{3}1$ allocation. While this assumption is not polynomial, the rest of Algorithm 4.7 runs in polynomial time. Thus a $\text{MMS}\frac{2}{3}1$ allocation can be computed in polynomial time with an oracle for MMS values. □

C.2 Proof of Theorem 4

Theorem 4. *For $n < 9$, Algorithm 4.7 computes a $(\frac{2}{3}, 1)$-MMS allocation in polynomial time.*

Proof. We follow the steps of Algorithm 4.7; however, we scale the instance so that $\text{MMS}_i = 1$. By Lemma 5, we can apply valid reductions and guarantee that any agent allocated a bundle during these steps receives value at least $v_i(A_i) \geq 1 \geq \text{MMS}_i^n$. We next observe that by Lemma 3, unordering an allocation results in an allocation for the original instance where every agent receives at least as much value as in the ordered allocation. Therefore, we focus on ordered instances where no valid reduction is possible; that is, $v_i(g_1) < 1$, $v_i(\{g_n, g_{n+1}\}) < 1$, and $v_i(M) = n$. Observe that because $v_i(g_n) \geq v_i(g_{n+1})$ and $v_i(\{g_n, g_{n+1}\}) < 1$, $v_i(g_{n+1}) < \frac{1}{2}$. Thus there are at most n high-value goods.

Since the instance is ordered, relabeling the goods on line 7 and initializing bag B to contain the good g_1 on line 9 guarantees that each time a bag is allocated, it contains at most one high-value good. This initialized bag cannot be allocated on its own because $v_i(g_1) < 1$ for all $i \in N'$. Each good added to B from

L on line 11 has value less than $\frac{1}{2}$ to all agent. Likewise, $v_i(B) < 1$ for all agents before the last good is added to B. Thus any bundle allocated on line 12 has value at most $\frac{3}{2}$ to all agents still in the market.

The bag-filling steps on lines 11 and 12 will continue to while there are sufficiently many low value goods to fill a bundle (this is checked on line 10). Suppose bag-filling stops after allocating $r = \lfloor \frac{2n}{3} \rfloor - \ell$ bundles. Let agent i be an agent in N' with the maximum number of high-value goods. Since the instance is ordered, this means that agent i values g_1 to g_h at least $\frac{1}{2}$. Under the original labeling of the goods at line 5, the first r items have been allocated in separate bags.

If $\ell \leq 0$, then the algorithm will terminate on line 18 resulting in an MMS$\frac{2}{3}1$ allocation. Suppose $\ell > 0$ Since agent i values the allocated bundles at no more than $\frac{3}{2}$, agent i must value the remaining unallocated goods least $n - \frac{3}{2}(\lfloor \frac{2n}{3} \rfloor - \ell) = (n - \frac{3}{2}\lfloor \frac{2n}{3} \rfloor) + \frac{3}{2}\ell \geq \frac{3}{2}\ell$.

If $h < \lfloor \frac{2n}{3} \rfloor$, then there are at most ℓ unallocated high-value goods each of value at most 1. Thus the remaining low-value goods have total value at least $\frac{\ell}{2}$. If $g_{r+1} \in H$ then agent i values g_{r+1} and the remaining low-value goods at least 1. If $g_{r+1} \in L$, then the remaining $\frac{3\ell}{2}$ value is contained in low-value goods. Thus a subset of these goods can be allocated to agent i. In either case, allocating a bundle to agent i contradicts that bag-filling stopped. Consider the following cases which depend on h, s, and ℓ.

1. Suppose $\ell \geq 2s$. Since the instance is ordered, the remaining $(h-r)$ high-value goods all have value at most $v_i(g_{r+1})$. For agent i, the remaining low-value goods have value at least $\frac{3}{2}\ell - v_i(g_{r+1})(h-r) = \frac{3}{2}\ell - v_i(g_{r+1})(s+\ell) \geq \frac{3}{2}\ell - v_i(g_{r+1})(\frac{3}{2}\ell) = \frac{3}{2}\ell(1 - v_i(g_{r+1})) \geq 1 - v_i(g_{r+1})$. Thus agent i values g_{r+1} together with the remaining low-value goods at least 1. This contradicts that bag-filling stopped before agent i received a bundle.
2. Suppose $\ell \leq s$. In this case, we observe that some agent in the market has high-value for the first and last high-value goods g_1 and g_h (after relabeling the goods on line 7). As in the proof of Theorem 3, all agents who agrees with this has at 2ℓ goods that they value at least $\frac{1}{2}$. Any agent who does not value $\{g_1, g_h\}$ at least 1 may consider g_h as a low-value good in later iterations of bag-filling. We observe that if the set of high-value goods does not change (because some other agent values g_{h-1} at least $\frac{1}{2}$) after undoing temporary allocations on line 17, then the set of allocated bundles in the next iteration of bag-filling will be identical to those allocated in the current round. While this may slow down the bag-filling procedure practically, it does not add more than a polynomial amount of work to the algorithm.
3. The remaining cases arise when $s < \ell < 2s$. We enumerate each case to show that they do not occur when $n < 9$. We consider cases as a tuple of the form (n, s, ℓ) for clarity. Since $n < 9$, and $s < \ell < 2s$, there are 7 cases.

   The cases $(5, 2, 3)$, $(7, 3, 4)$, and $(8, 3, 5)$ assume that no bundles are allocated during bag-filling. However, combining the facts that $v_i(g) \leq v_i(g_1) < 1$, there are at most n high-value goods, and $v_i(M) = n$, there must be at least $n - n(v_i(g_1)) \geq n(1 - v_i(g_1)) \geq 1 - v_i(g_1)$ value from low-value goods. Thus at least one bundle is allocated during bag-filling and these cases do not occur.

   The cases $(6, 2, 3)$ and $(8, 3, 4)$ only occur if some remaining agent i believes there are n high-value goods. Since $v_i(\{g_n, g_{n+1}\}) < 1$ all low value goods are worth less than $1 - v_i(g_n)$. If $\{g_1, g_{n+1}\}$ is worth value 1 to some agent, then the first bundle allocated was $\{g_1, g_{n+1}\}$. In this case, the amount

of value remaining from low-value goods is at least:

$$\begin{aligned}
&n - (v_i(g_1) + v_i(g_{n+1})) - \sum_{k=2}^{n} v_i(g_k) \\
&= n - (v_i(g_n) + v_i(g_{n+1})) - \sum_{k=1}^{n-1} v_i(g_k) \\
&> n - 1 - \sum_{k=1}^{n-1} v_i(g_k) \\
&\geq (1 - v_i(g_2)) + (n-2) - \sum_{k=1,k\neq 2}^{n-1} v_i(g_k) \\
&> (1 - v_i(g_2)) + (n-2) - (n-2) \\
&> 1 - v_i(g_2)
\end{aligned}$$

If $\{g_1, g_{n+1}\}$ is not worth 1 to any agent, then because low-value goods are added in descending order, the maximum value from low-value goods allocated in the first bundle is $2(1 - v_i(g_1))$. Thus the value remaining from low-value goods is at least:

$$\begin{aligned}
&n - \sum_{k=1}^{n} v_i(g_k) - 2(1 - v_i(g_1)) \\
&\geq (n-2) - \sum_{k=2}^{n-1} v_i(g_k) + (1 - v_i(g_1)) + (1 - v_i(g_n) \\
&\qquad - 2(1 - v_i(g_1)) \\
&\geq (n-2) - \sum_{k=2}^{n-1} v_i(g_k) \\
&= (1 - v_i(g_2)) + (n-3) - \sum_{k=3}^{n-1} v_i(g_k) \\
&\geq (1 - v_i(g_2)) + (n-3) - (n-3) \\
&= 1 - v_i(g_2)
\end{aligned}$$

In either case, there is at least enough value from low-value goods to fill another bundle during bag-filling contradicting that bag-filling stopped. Thus the cases $(6, 2, 3)$ and $(8, 3, 4)$ do not occur.
The remaining two cases are $(7, 2, 3)$ and $(8, 2, 3)$. In these cases, there are $n - 1$ high-value goods. We observe that the amount of value from low-value goods in these cases is at least:

$$
\begin{aligned}
& n - \sum_{k=1}^{n-1} v_i(g_k) \\
& \geq 1 + (n-1) - v_i(g_1) - v_i(g_2) - v_i(g_3) - \sum_{k=4}^{n-1} v_i(g_k) \\
& = 1 + (1 - v_i(g_1)) + (1 - v_i(g_2)) + (1 - v_i(g_3)) \\
& \qquad + (n-4) - \sum_{k=4}^{n-1} v_i(g_k) \\
& > 1 + (1 - v_i(g_1)) + (1 - v_i(g_2)) + (1 - v_i(g_3)) \\
& \qquad + (n-4) - (n-4) \\
& > (\frac{3}{2} - v_i(g_1)) + (\frac{3}{2} - v_i(g_2)) + (1 - v_i(g_3))
\end{aligned}
$$

Since the first two bundles have total value at most $\frac{3}{2}$, there is at least enough value to fill the first two bundles and still have enough low-value goods to fill a third bundle. This contradicts that only one or two bundles were filled during bag-filling (based on the cases). Thus neither cases $(7, 2, 3)$ nor $(8, 2, 3)$ occur.

□

D Optimal-MMS Fails for Chores

We begin with the generalized version of Theorem 5. We observe that when $d = \lfloor \frac{n}{2} \rfloor$ the following theorem becomes Theorem 1.

Theorem 8. *For any $n \geq 4$ and any $d \leq \lfloor \frac{n}{2} \rfloor$, there exists an instance with $O(dn)$ chores where any optimal-MMS allocation guarantees at most $\lceil \frac{n}{d} \rceil + 1$ agents their MMS value.*

Our construction is broken into three components: a tensor S, a tensor T, and d groups of agents. The entries of tensor S are chosen such that equi-partitions of the entries correspond to a set of slices of S. An equi-partition is a partition of the entries of a tensor where each set of the partition has the same sum. Here, all equi-partitions refer to n-partitions. Tensor T provides perturbation so that the only equi-partitions of $S + T$ are aligned slices.

All tensors in our construction are order d with dimensions $(n \times n \times \ldots \times n)$.[12] We say that tensor S is indexed by a d-tuple $(x_1, \ldots, x_d)$ corresponding to entry $S[x_1, \ldots, x_d]$ or $S_{x_1,\ldots,x_d}$ for short. The $(d-1)$-order slices of tensor S along dimension j are given by $S_j(1)$ to $S_j(n)$ where $S_j(i) = \{S_{x_1 \ldots x_d} \mid x_j = i\}$. We also define $S_{i;x_j=k}$ to be the entry where all indices are i except index $x_j = k$. For example, $S_{i;x_1=n} = S_{nii\ldots i}$. Tensor addition is entry-wise $((S+T)_{x_1,x_2,\ldots,x_d} = S_{x_1,x_2,\ldots,x_d} + T_{x_1,x_2,\ldots,x_d})$. Unless otherwise specified, the entries of a tensor have value 0.

Figure 9a depicts the indexing of an order 3 tensor with dimensions $(6 \times 6 \times 6)$. Figure 9b illustrates the distinct sets of slices of such a tensor. Lastly, Figure 9c depicts the non-zero entries of tensors S and T in our construction when $n = 6$ and $d = 3$.

Construction of Tensor S. Let S be an order d tensor whose non-zero entries are as follows:

- For $1 \leq i < n$ and $j \in [d]$, $S_{ii\ldots i} = -\frac{d^{n-i}-1}{d^{n-i}}$
- For $1 \leq i < n$ and $j \in [d]$, $S_{i;x_j=n} = -\frac{1}{d-1} \cdot \frac{1}{d^{n-i}}$
- $S_{nn\ldots n} = -1 + \sum_{i=1}^{n-1} \frac{1}{d-1} \cdot \frac{1}{d^{n-i}})$

Lemma 24. *Each $(d-1)$-order slice $S_i(j)$ has sum -1.*

Proof. Consider the non-zero entries within slice $S_j(i)$. First when $i \neq n$, the only non-zero entries of $S_j(i)$ are $S_{ii\ldots i}$ and $S_{i;x_k=n}$ for each $k \neq j$. Thus:

$$\begin{aligned}
\sum_{x \in S_j(i)} x &= S_{ii\ldots i} + \sum_{k \neq j}^{d} S_{i;x_k=n} \\
&= -\frac{d^{n-i}-1}{d^{n-i}} + \sum_{k \neq j}^{d} -\frac{1}{d-1} \cdot \frac{1}{d^{n-i}} \\
&= -\left(\frac{d^{n-i}-1}{d^{n-i}} + (d-1)\frac{1}{d-1} \cdot \frac{1}{d^{n-i}} \right) \\
&= -\left(\frac{d^{n-i}-1}{d^{n-i}} + \frac{1}{d^{n-i}} \right) \\
&= -1
\end{aligned}$$

[12] When referring to a tensor, the "order" is the number of dimensions that the tensor is embedded in. The "dimensions" of a tensor, however, refers to the number of entries along each axis, similar to the dimensions of a matrix.

When $i = n$, the non-zero entries of slice $S_j(i)$ are $S_{nn\ldots n}$ and $S_{k;x_j=n}$ for each $k \neq n$. Thus:

$$\begin{aligned}\sum_{x \in S_j(n)} x &= S_{nn\ldots n} + \sum_{k \neq n}^{n} S_{k;x_j=n} \\ &= -1 + \sum_{i=1}^{n-1} \frac{1}{d-1} \cdot \frac{1}{d^{n-i}} + \sum_{k=1}^{n-1} -\frac{1}{d-1} \cdot \frac{1}{d^{n-k}} \\ &= -1\end{aligned}$$

□

The combinatorial structure of S has a few other desirable properties. First observe that the entries on the main diagonal ($S_{ii\ldots i}$) all have value less than $-\frac{1}{2}$. When considering equi-partitions of the entries of S, these entries must all be included in separate sets. Furthermore, the entries off the main diagonal increase when approaching $S_{nn\ldots n}$ in such a way that all d of the entries with value $-\frac{1}{d-1} \cdot \frac{1}{d^{n-i}}$ are required to account for a single entry with value $-\frac{1}{d-1} \cdot \frac{1}{d^{n-(i+1)}}$.

Lemma 25. *The sets of slices $\{S_{j_1}(1), S_{j_2}(2), \ldots, S_{j_n}(n)\}$ are the only equi-partitions equi-partition of the non-zero entries of S.*

Proof. Lemma 24 implies that a set of slices of the form $\{S_{j_1}(1), S_{j_2}(2), \ldots, S_{j_n}(n)\}$ forms an equi-partition. In order to show that these equi-partitions are unique, consider the set which contains $S_{11\ldots1}$. Since $-1 + S_{11\ldots1} = -\frac{1}{d^{n-1}} > -\frac{1}{d-1} \cdot \frac{1}{d^{n-2}}$. Thus the only entries which can be included with $S_{11\ldots1}$ are of the form $-\frac{1}{d-1} \cdot \frac{1}{d^{n-1}}$. Importantly, $d-1$ of these entries uniquely identify a $(d-1)$-order slice $S_{j_1}(1)$ where $S_{1;x_{j_1}=n}$ is the missing entry with value $-\frac{1}{d-1} \cdot \frac{1}{d^{n-1}}$.

By induction, with a similar argument at each step, each successive entry on the main diagonal must be included in a set which corresponds exactly to a slice $S_{j_i}(i)$. □

Construction of Tensor T. In Lemma 25, the slices of S are not necessarily aligned slices. For example, in the case of three dimensions, the first slice might be frontal while the second one might be vertical. Since all slices have the same sum, there is no reason an agent wouldn't compute MMS by taking slices at random. In order to rectify this, we construct tensor T so that equi-partitions of $S + T$ correspond only to aligned slices.

Let tensor T be an order d tensor with non-zero values as follows:

- For $1 \leq i < n-1$, $j \in [d]$, $T_{i;x_j=n} = r_{ij}$
- For $1 \leq i < n-1$, $j \in [d]$, $T_{i;x_j=i+1} = -u_{ij}$
- For $j \in [d]$, $T_{n-1;x_j=n} = -x_j$
- $T_{nn\ldots n} = -z_1$

Where

- $r_{ij} = \epsilon^{d(n-1)-di-j+1}$
- $u_{1j} = r_{1j}$
- $u_{ij} = \frac{1}{d-1}\left(\sum_{k\neq j}^{d} -u_{i-1,k}\right) + \frac{d-2}{d-1}u_{i-1,j} + r_{ij} \approx r_{ij}$
- $x_j = u_{n-1,j} - r_{n-1,j}$
- $z_j = \left(\sum_{i=1}^{n-2} r_{ij}\right) - x_j$

It is important to observe that the entries of T are all proportional to powers of a very small ϵ term. While some of these values are positive, the positive entries only occur in indices where S has negative value. Thus if ϵ is small enough, $S + T$ has only non-positive values. Another important observation is that the slices of T all have sum 0.

Lemma 26. *Each $(d-1)$-order slice $T_j(i)$ has sum 0.*

Proof. We first address the case of $i = 1$. Slice $T_j(1)$ contains the non-zero entries $T_{1;x_k=n}$ for each $k \neq j$ and $T_{1;x_k=2}$ for each $k \neq j$. For each $k \neq j$, the entry $T_{1;x_k=n} = r_{1j}$ and $T_{1;x_k=2} = -u_{1j}$. Since $-u_{1j} = r_{1j}$ the sum of these two entries is 0. Adding over all $k \neq j$ yields that the first slice (independent of j) has sum 0.

Now consider slice $T_j(i)$ for $1 < i < n-1$. The non-zero entries of T in this slice are $T_{i;x_k=n}$ and $T_{i;x_k=i+1}$ for $k \neq j$ and $T_{i-1;x_j=i}$. Adding these entries together yields:

$$
\begin{aligned}
&\sum_{x \in T_j(i)} x \\
&= \sum_{k \neq j}^{d} T_{i;x_k=i+1} + \sum_{k \neq j}^{d} T_{i;x_k=n} + T_{i-1;x_j=i} \\
&= \sum_{k \neq j}^{d} (-u_{ik}) + \sum_{k \neq j}^{d} (r_{ik}) - u_{i-1,j} \\
&= \sum_{k \neq j}^{d} - \left(\frac{1}{d-1} \left(\sum_{l \neq k}^{d} -u_{i-1,l} \right) + \frac{d-2}{d-1} u_{i-1,k} + r_{ik} \right) \\
&\quad + \sum_{k \neq j}^{d} (r_{ik}) - u_{i-1,j} \\
&= \sum_{k \neq j}^{d} \left(\frac{1}{d-1} \left(\sum_{l \neq k}^{d} u_{i-1,l} \right) - \frac{d-2}{d-1} u_{i-1,k} - r_{ik} \right) \\
&\quad + \sum_{k \neq j}^{d} (r_{ik}) - u_{i-1,j} \\
&= \sum_{k \neq j}^{d} \left(\frac{1}{d-1} \left(\sum_{l \neq k}^{d} u_{i-1,l} \right) - \frac{d-2}{d-1} u_{i-1,k} \right) - u_{i-1,j} \\
&= \frac{1}{d-1} \left(\sum_{k \neq j}^{d} \sum_{l \neq k}^{d} (u_{i-1,l}) - (d-2) \sum_{k \neq j}^{d} (u_{i-1,k}) \right) - u_{i-1,j} \\
&= \frac{1}{d-1} \left((d-1)(u_{i-1,j}) \right) - u_{i-1,j} = 0
\end{aligned}
$$

The last step follows because each of the $(d-1)$ summations (for each $k \neq j$) adds a single $u_{i-1,j}$ term. For each $k \neq j$, $(d-2)$ of the summations (all but $l = k$ of the $(d-1)$ inner summations) add a $u_{i-1,k}$ term.

We next address the case where $i = n-1$. In this case, we observe that $i+1 = n$ so the only entries in

this slice are $T_{n-1;x_k=n}$ for $k \neq j$ and $T_{n-2;x_j=n-1}$. Adding these entries together yields:

$$\begin{aligned}
&\sum_{x \in bi T_j(n-1)} x \\
&= \sum_{k \neq j}^{d} T_{n-1;x_k=n} + T_{n-1;x_j=n-1} \\
&= \sum_{k \neq j}^{d} (-x_k) - u_{n-2,j} \\
&= \sum_{k \neq j}^{d} -\left(u_{n-1,k} - r_{n-1,k}\right) + u_{n-2,j} \\
&= \sum_{k \neq j}^{d} -\Bigg(\Bigg(\frac{1}{d-1}\left(\sum_{l \neq k}^{d} -u_{n-2,l}\right) \\
&\quad + \frac{d-2}{d-1} u_{i-2,k} + r_{n-1,k}\Bigg) - r_{n-1,k}\Bigg) - u_{n-2,j} \\
&= \sum_{k \neq j}^{d} \left(\frac{1}{d-1}\left(\sum_{l \neq k}^{d} u_{n-2,l}\right) - \frac{d-2}{d-1} u_{i-2,k}\right) - u_{n-2,j} \\
&= \frac{1}{d-1}\left(\sum_{k \neq j}^{d} \sum_{l \neq k}^{d} \left(u_{n-2,l}\right) - (d-2) \sum_{k \neq j}^{d} \left(u_{i-2,k}\right)\right) - u_{n-2,j} \\
&= \frac{1}{d-1}((d-1)(u_{n-2,j})) - u_{n-2,j} = 0
\end{aligned}$$

We last address the case where $i = n$. In this case, the non-zero entries in this slice are $T_{i;x_j=n}$ for each $i \in [n-2]$, $T_{n-1;x_j=n}$, and $T_{nn\ldots n}$. Adding these entries together yields:

$$\begin{aligned}
\sum_{x \in T_j(n)} x &= \sum_{k=1}^{n-2} T_{k;x_j=n} + T_{n-1;x_j=n} + T_{nn\ldots n} \\
&= \sum_{k=1}^{n-2} \left(r_{kj}\right) - x_j - z_1
\end{aligned}$$

If $j = 1$ then the term $\sum_{k=1}^{n-2} (r_{kj}) - x_j = z_1$ so this trivially sums to 0. If $j \neq 1$, then we first observe that the sum of all entries of T is 0 by adding the sums of the slices $\{T_1(1), T_1(2), \ldots, T_1(n)\}$. Since $\sum_{i=1}^{n} \sum_{x \in T_j(i)} x = 0$ and $\sum_{k=1}^{n-1} \sum_{x \in T_j(k)} x = 0$, subtracting the two equations gives $\sum_{x \in T_j(n)} = 0$. □

Combining Lemmas 25 and 26 together immediately shows that any set of aligned slices of $(S+T)$ forms an equi-partition of the non-zero entries where each set has sum -1. We now show that these partitions are unique. For intuition, the key point in our construction is that $(d-1)$ of the $T_{i;x_j=i+1}$ terms all lie in the same slice $T_j(i)$. Once j has been chosen for $T_j(1)$ the extra term $T_{1;x_j=2}$ lies in slice $T_j(2)$.

Lemma 27. *The only equi-partitions of the non-zero entries of $(S+T)$ correspond to aligned slices.*

Proof. First observe that if ϵ is sufficiently small then the structure of Lemma 25 still holds. Consider first the set of an equi-partition which contains $(S+T)_{11\ldots1}$. In order for this set to have value -1, it must have $(d-1)$ of the entries which receive $-\frac{1}{d-1} \cdot \frac{1}{d^{n-1}}$ value from S. Again, selecting $(d-1)$ of these uniquely characterizes a slice $(S+T)_j(1)$. Observe that these entries have positive perturbations from T corresponding to r_{1k} for each $k \neq j$. In order for these perturbations to be offset so that the set containing $(S+T)_{11\ldots1}$ has

sum -1, the corresponding entries with values $-u_{1k}$ for $k \neq j$ must be included in this set. These entries correspond exactly to the slice $(S+T)_j(1)$.

As in the proof of Lemma 25, we observe that the entries on the main diagonal $(S+T)_{ii\ldots i}$ must be in separate sets. Likewise, because ϵ is small, a similar argument enforces that $(d-1)$ of the entries with value $-\frac{1}{d-1} \cdot \frac{1}{d^{n-i}}$ must be included in this set. We address the positive perturbations (from T) of these entries by induction. Observe that the smallest perturbations from T occur in the earliest slices ($(1,1) = \arg\max_{i,j} -r_{ij}$). Note also that $(-u_{1j}, r_{ij})$ is the only pair of perturbations from T which are excluded from slice $(S+T)_j(1)$. A similar argument for slice $(S+T)_j(1)$ holds for slice $(S+T)_j(2)$: in order for the set containing $(S+T)_{22\ldots2}$ to have value -1, it must also have $(d-1)$ entries with the r_{2k} perturbations from T. Since $\epsilon^2 \ll \epsilon$, each r_{2k} entry can only be offset with entries with perturbation $-u_{2k}$. (All but one of the smaller perturbation entries have already been assigned to slice 1.) However, by Lemma 26, we see that adding together all $k \neq l$ entries of the form r_{2k} and $-u_{2k}$ leaves value $-u_{1l}$ (since $-u_{1l} + \sum_{k\neq l}^{d} -u_{2k} + \sum_{k\neq l}^{d} r_{2k} = 0$). Observe that only a single entry of the form $-u_{1l}$ is available: notably $-u_{1j}$. Thus the set which contains $(S+T)_{22\ldots2}$ corresponds to the slice $(S+T)_j(2)$.

By induction on this same argument we see that each set containing the entry $(S+T)_{ii\ldots i}$ for $1 \leq i < n$ must correspond to a slice $(S+T)_j(i)$ for some fixed j (after j is chosen for the first slice). We conclude this proof by observing that the remaining entries all lie in the last slice $(S+T)_j(n)$ and have total sum -1 by Lemmas 25 and 26. □

With the details of S and T, we are now ready to prove the main theorem. We restate it here for convenience.

Theorem 8. *For any $n \geq 4$ and any $d \leq \lfloor \frac{n}{2} \rfloor$, there exists an instance with $O(dn)$ chores where any optimal-MMS allocation guarantees at most $\lceil \frac{n}{d} \rceil + 1$ agents their MMS value.*

Proof. Lemma 27 implies that there are d unique equi-partitions of the entries of $(S+T)$. We group agents arbitrarily into d groups of size at least 2 and at most $\lceil \frac{n}{d} \rceil$ each. We next show how to perturb $(S+T)$ for each group so that each group has a unique equi-partition. Furthermore, we enforce that the only entry with positive perturbation for all agents is the last entry on the main diagonal. Let P^j be an order d tensor with non-zero values as follows:

- For $1 \leq i < n$, $P^j_{i;x_j=n} = -\tilde{\epsilon}$
- $P^j_{nn\ldots n} = (n-1)\tilde{\epsilon}$

The entry where P^j is positive correspond to entries where $(S+T)$ has negative value. Thus $(S+T+P^j)$ has only non-positive entries if $\tilde{\epsilon}$ is sufficiently small. Observe that the slices $\{(S+T+P^j)_j(1), (S+T+P^j)_j(2), \ldots, (S+T+P^j)_j(n)\}$ form an equi-partition of the entries of $(S+T+P^j)$ as P^j has non-zero values only in the last slice $(S+T+P^j)_j(n)$ and the total sum of its non-zero entries is 0. If $\tilde{\epsilon}$ is sufficiently small (so that the structure of $S+T$ dominates when creating an equi-partition), then none of the other sets of slices of $(S+T+P^j)$ forms an exact equi-partition as every slice except for the last contains exactly one of the $-\tilde{\epsilon}$ perturbations.

We now create an instance $I = \langle N, M, V \rangle$ where M contains one good for each non-zero entry of $(S+T)$. Agents in group j value the goods according to the values of $(S+T+P^j)$. Since every agent is able to form an equi-partition of the entries of their valuation tensor, all agents have $\text{MMS}_i = -1$. By allocating goods according to slices, each agent receives $(-1-\tilde{\epsilon})$ value which implies the optimal-MMS constant is at most $(-1-\tilde{\epsilon})$. Since $\tilde{\epsilon}$ is very small, the only way all agents can receive at most $-1-\tilde{\epsilon}$ value is if the chores are allocated according to aligned slices (Lemma 27). Under this allocation, at most one group and a single

additional agent who receives the last slice (since $(S+T+P^j)_j(n)$ has positive perturbation $(n-1)\tilde{\epsilon}$) receive their MMS value. Lastly we observe that there only $O(dn)$ goods in M. $\square$

Again we note that if $d = \lfloor \frac{n}{2} \rfloor$ in the above theorem, then we see that only 3 agents (4 if n is odd) receive their full MMS in any optimal-MMS allocation.