Ordinal Maximin Share Approximation for Chores

Hadi Hosseini
Pennsylvania State University
hadi@psu.edu

Andrew Searns
Johns Hopkins University Applied Physics Laboratory
andrew.searns@jhuapl.edu

Erel Segal-Halevi
Ariel University
erelsgl@gmail.com

ABSTRACT

We study the problem of fairly allocating a set of m indivisible chores (items with non-positive value) to n agents. We consider the desirable fairness notion of 1-out-of-d maximin share (MMS)—the minimum value that an agent can guarantee by partitioning items into d bundles and receiving the least valued bundle—and focus on ordinal approximation of MMS that aims at finding the largest $d \leq n$ for which 1-out-of-d MMS allocation exists. Our main contribution is a polynomial-time algorithm for 1-out-of-$\lfloor \frac{2n}{3} \rfloor$ MMS allocation, and a proof of existence of 1-out-of-$\lfloor \frac{3n}{4} \rfloor$ MMS allocation of chores. Furthermore, we show how to use recently-developed algorithms for bin-packing to approximate the latter bound up to a logarithmic factor in polynomial time.

KEYWORDS

Fair Division; Maximin Share Guarantee; Resource Allocation

ACM Reference Format:
Hadi Hosseini, Andrew Searns, and Erel Segal-Halevi. 2022. Ordinal Maximin Share Approximation for Chores. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

Fairness is one of the most fundamental requirements in many multiagent systems. Fair division, in particular, deals with allocation of resources and alternatives in a fair manner by cutting across a variety of fields including computer science, economics, and artificial intelligence. Traditionally, fair division has been concerned with the allocation of *goods* that are positively valued by agents, leading to a plethora of fairness notions, axiomatic results, and computational studies (see [18] and [45] for detailed discussions). However, many practical problems require the distribution of a set of negatively valued items (aka *chores*). These problems range from assigning household chores or distributing cumbersome tasks to those involving collective ownership responsibility [48] in human-induced factors such as climate change [52], nuclear waste management, or controlling gas emissions [20]. The problem of allocating chores is crucially different from allocating goods both from axiomatic and computational perspectives. For instance, while goods are freely disposable, chores must be completely allocated. These fundamental differences have motivated a large number of recent works in fair division of divisible [15, 21] and indivisible chores [3, 5, 9, 29].

When dealing with indivisible items, a compelling fairness notion is the Maximin Share (MMS) guarantee—proposed by Budish [19]—which is a generalization of the *cut-and-choose protocol* to indivisible items [17]. An agent's 1-out-of-d maximin share value is the value that it can guarantee by partitioning m items into d bundles and receiving the least valued bundle. Unfortunately, the 1-out-of-n MMS allocations may neither exist for goods [28, 43] nor for chores [9]. These non-existence results, along with computational intractability of computing such allocations, have motivated *multiplicative* approximations of MMS wherein each agent receives an $\alpha \leq 1$ fraction of its 1-out-of-n MMS value when dealing with goods [30–32], or $\alpha \geq 1$ approximation of its 1-out-of-n MMS value when dealing with chores [9, 13, 39].

In this paper, we initiate the study of *ordinal* MMS approximations for allocating chores. The goal is finding an integer $d \leq n$ for which 1-out-of-d MMS exists and can be computed efficiently. Recently, ordinal approximations of MMS for allocating 'goods' have received particular attention as natural guarantees that provide a simple conceptual framework for justifying approximate decisions to participating agents: partition the items in a counterfactual world where there are $d \geq n$ agents available [10, 11, 27, 36, 51]. Since these approximations rely on ordinal rankings of bundles, they are generally robust against slight changes in agent's valuation profiles compared to their multiplicative counterparts (see [38] for an example and a detailed discussion). Focusing on ordinal approximations, we discuss key technical differences between allocating goods and chores, and highlight practical computational contrasts between ordinal and multiplicative approximations of MMS.

1.1 Contributions

We make the following theoretical and algorithmic contributions.

An algorithm for **1*-out-of-*$\lfloor \frac{2n}{3} \rfloor$ *MMS.*** We show that heuristic techniques for allocating goods do not carry over to chores instances (Section 3), and develop other techniques to upper-bound the number of large chores (Lemma 3.2). Using these techniques, we develop a greedy algorithm that achieves 1-out-of-$\lfloor \frac{2n}{3} \rfloor$ MMS approximation for chores (Theorem 4.1). The algorithm runs in strongly-polynomial time: the number of operations required is polynomial in the number of agents and chores.

Existence of **1*-out-of-*$\lfloor \frac{3n}{4} \rfloor$ *MMS.*** We show the existence of 1-out-of-$\lfloor \frac{3n}{4} \rfloor$ MMS allocation of chores (Theorem 5.1). The main technical challenge is dealing with large chores which requires exact computation of MMS values, rendering our algorithmic approach intractable. While our technique gives the best known ordinal approximation of MMS, it only provides a tight bound for small instances (Example 5.1) but not necessarily for larger instances (Proposition 5.1).

Efficient approximation algorithm. We develop a practical algorithm for approximating the 1-out-of-$\lfloor \frac{3n}{4} \rfloor$ MMS bound for chores. More specifically, our algorithm guarantees 1-out-of-d MMS

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

for $d = \lfloor \lfloor \frac{3n}{4} \rfloor - O(\log n) \rfloor$ (Theorem 6.1) and runs in time polynomial in the binary representation of the input.

1.2 Related Work

MMS for allocating goods. The notion of maximin-share originated in the economics literature. Budish [19] showed a mechanism that guarantees 1-out-of-$(n+1)$ MMS to all agents by adding a small number of excess goods. Whether or not 1-out-of-$(n+1)$ MMS can be guaranteed without adding excess goods remains an open problem to date.

In the standard fair division settings, in which adding goods is impossible, the first non-trivial ordinal approximation was 1-out-of-$(2n-2)$ MMS [1]. Hosseini and Searns [36] studied the connection between guaranteeing 1-out-of-n MMS for 2/3 of the agents and the ordinal approximations for *all* agents. The implication of their results is the existence of 1-out-of-$(\lfloor 3n/2 \rfloor)$ MMS allocations and a polynomial-time algorithm for $n < 6$. Recently, a new algorithmic method has been proposed that achieves this bound for any number of agents [37]. The ordinal approximations have been extended to ℓ-out-of-d MMS to guarantee that each agent receives at least as much as its worst ℓ bundles, where the goods were partitioned into d bundles [10, 50]. The maximin share and its ordinal approximations have also been applied to some variants of the *cake-cutting* problem [14, 26, 27].

The multiplicative approximation of MMS originated in the computer science literature [47]. These algorithms guarantee that each agent receives at least an α fraction of its maximin share threshold [2, 30, 32, 43]. For goods, the best known existence result is $\alpha \geq 3/4 + 1/(12n)$, and the best known polynomial-time algorithm guarantees $\alpha \geq 3/4$ [31]. The MMS bound was improved for special cases with only three agents [2], and the best known approximation is $\alpha \geq 8/9$ [33].

There are also MMS approximation algorithms for settings with constraints, such as when the goods are allocated on a cycle and each agent must get a connected bundle [53]. McGlaughlin and Garg [44] showed an algorithm for approximating the maximum Nash welfare (the product of agents' utilities), which attains a fraction $1/(2n)$ of the MMS. Recently, Nguyen et al. [46] gave a Polynomial Time Approximation Scheme (PTAS) for a notion defined as *optimal-MMS*, that is, the largest value, α, for which each agent receives the value of $\alpha \cdot \text{MMS}_i$. Since the number of possible partitions is finite, an optimal-MMS allocation always exists, and it is an MMS allocation if $\alpha \geq 1$. However, an optimal-MMS allocation may provide an arbitrarily bad ordinal MMS guarantee [36, 49].

MMS for allocating chores. Aziz et al. [9] initiated the study of MMS fairness for allocating indivisible chores. They proved that—similar to allocating goods—a 1-out-of-n MMS allocation may not always exist, and computing the MMS value for a single agent remains NP-hard.

In the maximin share allocation of chores, the multiplicative approximation factor is larger than 1 (each agent might get a larger set of chores than its MMS value). The multiplicative factors in the literature have been improved from 2 [9] to 4/3 [13] to 11/9 [39]. The best known polynomial-time algorithm guarantees a 5/4 factor [39]. Aigner-Horev and Segal-Halevi [1] prove the existence of a 1-out-of-$\lfloor 2n/3 \rfloor$ MMS allocation for chores, but their algorithm requires an exact computation of the MMS values, so it does not run in polynomial time. Note that multiplicative and ordinal approximations do not imply one another—each of them might be better in some instances as we illustrate in the next example.

Example 1.1. Consider an instance with $n = 3$ agents and m identical chores of value -1. Then:

- If there are $m = 2$ chores, then the 1-out-of-$\lfloor 2n/3 \rfloor$ MMS is -1, which is better than 11/9 of the 1-out-of-n MMS.
- If there are $m = 3$ chores, then the 1-out-of-$\lfloor 2n/3 \rfloor$ MMS is -2, which is worse than 11/9 of the 1-out-of-n MMS.

In the full version of the paper [38], we generalize this example to any number of agents. Additionally, we study the relationships between the ordinal maximin share and other common fairness notions such as approximate-proportionality or approximate-envy-freeness. The bottom line is that all these notions are independent: none of them implies a meaningful approximation of the other.

The notion of maximin share fairness has been extended to *asymmetric agents*, i.e. agents with different entitlements over chores [4, 5]. Recently, a variation of MMS has also been studied in conjunction with *strategyproofness* that only elicits ordinal preferences as opposed to cardinal valuations [6, 7]. In parallel, there are works studying other fairness notions for chores, or for combinations of goods and chores. Examples are approximate proportionality [8], approximate envy-freeness [3], approximate equitability [29], and leximin [22]. In the context of mixed items, however, no multiplicative approximation of MMS is guaranteed to exist [42]. In [38], we show that similarly no ordinal MMS approximation is guaranteed to exist for mixed items.

2 PRELIMINARIES

Problem instance. An instance of a fair division problem is denoted by $I = \langle N, M, V \rangle$ where $N = \{1, \ldots, n\}$ is a set of agents, $M = \{c_1, \ldots, c_m\}$ is a set of m indivisible chores, and $V = (v_1, \ldots, v_n)$ is a valuation profile of agents. Agent i's preferences over chores is specified by a valuation function $v_i : 2^M \rightarrow \mathbb{R}$. We assume that the valuation functions are *additive*; that is, for any agent $i \in N$, for each subset $S \subseteq M$, $v_i(S) = \sum_{c \in S} v_i(\{c\})$ where $v_i(\emptyset) = 0$. We assume items are chores for all agents, i.e., for each $i \in N$, for every $c \in M$ we have $v_i(\{c\}) \leq 0$. For a single chore $c \in M$, we write $v_i(c)$ instead of $v_i(\{c\})$. Without loss of generality, we assume that $m \geq n$ since otherwise we can add dummy chores that are valued 0 by all agents.

Allocation. An allocation $A = (A_1, \ldots, A_n)$ is an n-partition of the set of chores, M, where a bundle of chores A_i, possibly empty, is allocated to each agent $i \in N$. An allocation must be *complete*: $\cup_{i \in N} A_i = M$.

Maximin share. Let $d \leq n$ be an integer and $\Pi_d(M)$ denote the set of d-partitions of M. For each agent $i \in N$, the **1-out-of-*d* Maximin Share** of i on M, denoted $\text{MMS}_i^d(M)$, is defined as

$$\text{MMS}_i^d(M) = \max_{(A_1, A_2, \ldots A_d) \in \Pi_d(M)} \min_{j \in [d]} v_i(A_j),$$

where $[d] = \{1, \ldots, d\}$. Intuitively, this is the maximum value that can be guaranteed if agent i partitions the items into d bundles and chooses the least valued bundle. When it is clear from the context, we write MMS_i^d or 1-out-of-d MMS to refer to $\text{MMS}_i^d(M)$.

Given an instance, we say that a 1-out-of-d MMS exists if there exists an allocation $A = (A_1, \ldots, A_n) \in \Pi_n(M)$ such that for every agent $i \in N$, $v_i(A_i) \geq \text{MMS}_i^d(M)$. Note that $\text{MMS}_i^d(M) \leq \frac{v_i(M)}{d}$ and it is a weakly-increasing function of d: a larger d value means that there are more agents to share the burden, so each agent potentially has fewer chores to do. Clearly, $\text{MMS}_i^d = \frac{v_i(M)}{d}$ when chores can be partitioned into d bundles of equal value. Moreover, $\frac{v_i(M)}{n}$ is agent i's *proportional share.*

Ordered instance. An instance I is *ordered* when all agents agree on the linear ordering of the items, irrespective of their valuations. Formally, I is an *ordered instance* if there exists an ordering $(c_1, c_2, \ldots, c_m)$ such that for all agents $i \in N$ we have $|v_i(c_1)| \geq |v_i(c_2)| \geq \ldots \geq |v_i(c_m)|$. Throughout this paper, we often refer to this as an ordering from the *largest chores* (least preferred) to the *smallest chores* (most preferred).

In the context of allocating goods, Bouveret and Lemaître [16] introduced ordered instances as the 'most challenging' instances in achieving MMS, and showed that given an *un*ordered instance, it is always possible to generate a corresponding ordered instance in polynomial time.[1] More importantly, if an ordered instance admits an MMS allocation, the original instance also admits an MMS allocation which can be computed in polynomial time (see Example 2.1).

Lemma 2.1 (Barman and Krishna Murthy [13]). *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 $i \in N, v_i(A_i) \geq v'_i(A'_i)$.*

The above results hold for any MMS approximation without loss of generality, and have been adopted extensively in simplifying the MMS approximations of chores [39]. Therefore, throughout the paper we only focus on ordered instances.

Example 2.1 (Ordering an instance). Consider the following unordered instance with four chores and two agents:

	c_1	c_2	c_3	c_4	MMS_i^n	$v_i(A_i)$
a_1	-3	$\textcircled{-5}$	-6	$\textcircled{-1}$	-8	-6
a_2	$\textcircled{-2}$	-8	$\textcircled{-4}$	-9	-12	-6

An ordered instance is obtained by sorting the values in descending order of absolute values. It has two possible allocations marked by a circle and * that satisfy MMS:

	c'_1	c'_2	c'_3	c'_4	MMS_i^n
a_1	-6*	$\textcircled{-5}$	$\textcircled{-3}$	-1*	-8
a_2	$\textcircled{-9}$	-8*	-4*	$\textcircled{-2}$	-12

The marked MMS allocations in the ordered instance corresponds to a picking-sequence that results in an MMS allocation in the original instance. A picking sequence lets agents select items from the 'best chores' (most preferred) to the 'worst chores' (least preferred).

For instance, applying a picking sequence 2, 1, 1, 2 (obtained from the circled allocation in the second table) to the original instance results in allocation A (marked by circles in the first table) that guarantees MMS. Specifically, when applied to the original instance, agent 2 picks first, and takes its highest valued chore c_1, which corresponds to c'_4. Agent 1 then picks its best chore c_4, which is available. The next pick also belongs to agent 1. But his second-best chore is c_1, which is already allocated to agent 2. Thus, agent 1 picks its next-best available chore c_2, and agent 2 is left with c_3.

[1]Bouveret and Lemaître [16] called these same-order preferences.

3 VALID REDUCTIONS FOR CHORES

In this section, we first show that the valid reductions techniques that are typically used for allocating goods can no longer be applied to chores instances. While typical goods reductions fail in allocating chores, we then argue that some of the core ideas translate to chores allocation through careful adaptations. These techniques are of independent interest as they can be utilized in other heuristic algorithms (e.g. multiplicative MMS approximations).

3.1 Reductions for goods

Several algorithms that are developed to provide multiplicative MMS approximations rely on structural properties of MMS and heuristic techniques to avoid computational barriers of computing MMS thresholds. To understand common reduction techniques, we first take a detour to recall techniques that are valid when allocating goods. For the ease of exposition, we present this section with the standard definition of 1-out-of-n MMS.

Definition 3.1 (Valid Reduction for Goods). Given an instance, $I = \langle N, M, V \rangle$ and a positive integer n, allocating a set of goods $A_i \subseteq M$ to an agent $i \in N$ is a *valid reduction* if
(i) $v_i(A_i) \geq \text{MMS}_i^n(M)$, and
(ii) $\forall j \in N \setminus \{i\}, \text{MMS}_j^{n-1}(M \setminus A_i) \geq \text{MMS}_j^n(M)$.

Intuitively, a valid reduction ensures that the MMS values of the remaining agents in the reduced instance does not strictly decrease; otherwise, solving the reduced instance may violate the initial MMS values of agents.

Since computing MMS values is NP-hard [16], one can instead utilize proportionality as a (loose) upper bound for MMS values. Given the proportionality bound, it is easy to see that for each agent $i \in N, \text{MMS}_i^n(M) \leq \frac{v_i(M)}{n}$. Therefore, any good $g \in M$ with a value $v_i(g) \geq \frac{v_i(M)}{n}$ for agent i can be assigned to agent i, satisfying i's MMS value, without violating conditions of valid reductions. The next lemma (due to Garg et al. [30]) formalizes this observation and provides two simple reduction techniques.

Lemma 3.1 (Garg et al. [30]). *Given an ordered goods instance $I = \langle N, M, V \rangle$ with $|N| = n$, if $v_i(\{g_n, g_{n+1}\}) \geq \frac{v_i(M)}{n}$, then allocating $A_i = \{g_n, g_{n+1}\}$ to agent i (and removing them from the instance) forms a valid reduction. Similarly, allocating $\{g_1\}$ to agent i forms a valid reduction if $v_i(\{g_1\}) \geq \frac{v_i(M)}{n}$.*

Remark 3.1. When allocating goods, valid reduction techniques are often used together with scaling of an instance to simplify the approximation algorithms [30, 31]. The **scale invariance** property of MMS [32] states that if an agent's valuations are scaled by a factor, then its MMS value scales by the same factor. Formally, given an instance $I = \langle N, M, V \rangle$, for every agent $i \in N$ with a proportionality bound $\frac{v_i(M)}{n}$ we can construct a new instance $I' = \langle N, M, V' \rangle$ such that $v'_i(M) = n$ and for every $g \in M, v'_i(g) = \frac{n}{v_i(M)} v_i(g)$. Using the proportionality bound for scaling an instance implies that allocating any set $S \in M$ such that $v_i(S) \geq 1$ to agent i forms a valid reduction.

The scale invariance property of MMS and reduction techniques circumvent the exact computation of MMS thresholds, which enables greedy approximation algorithms for allocating goods. Garg et al. [30] developed a simple greedy algorithm that guarantees to each agent 2/3 of its MMS value; later algorithms improved this approximation to 3/4 [31, 32].

3.2 Failure of Goods Reductions

We briefly discuss how the valid reductions for goods do not translate to instances with chores. The reason is that the reductions for goods rely upon the fact that, redistributing items from one bundle of a partition to other bundles weakly increases the value of other bundles. However, in the context of chores, this assumption does not hold as we illustrate next.

Example 3.1. Consider three agents and six chores. Agents' valuations are identical such that each agent $i \in N$ values each chore $c \in M$ as $v_i(c) = -1$. The 1-out-of-3 MMS of all agents is -2, i.e. $\text{MMS}_i^3 = -2$ for every $i \in N$. A reduction that allocates a single chore (e.g. largest chore), say c_1, satisfies agent 1 since $v_1(c_1) = -1 \geq \text{MMS}_1^3$. However, this reduction is *not* valid since the MMS value of the remaining agents decreases, that is, $\text{MMS}_i^2 = -3$ for $i \in \{2, 3\}$.

To illustrate why reductions of larger bundles such as $\{c_n, c_{n+1}\}$ fail, we provide the following example that generalizes this reduction to bundles with larger sizes.

Example 3.2. Consider an instance with three agents and $3(k+2)$ chores that are each valued -1. Each agent's MMS value is $\text{MMS}_i^3 = -(k+2)$. Take any bundle $S \subset M$ of $k+1$ chores. Any agent i would agree to receive S, as $v_i(S) = -(k+1) \geq \text{MMS}_i^3 = -(k+2)$. However, allocating the bundle S to agent i is not a valid reduction. This is because the remaining $2k + 5$ chores must be allocated among the remaining two agents, but $\text{MMS}_j^2(M \setminus S) = -(k + 3)$ which is less than $\text{MMS}_j^3 = -(k + 2)$.

Notice that smaller bundles of $v_i(\{c_n, c_{n+1}\}) = -2$ do satisfy agent i as well but still result in decrease of MMS values for other agents. For example, when $k = 2$, if $\{c_3, c_4\}$ are allocated to an agent, the MMS values of the remaining agents decrease from $\text{MMS}_i^3 = -4$ to $\text{MMS}_i^2 = -5$.

3.3 Estimating the Number of Large Chores

One of the key distinctions between allocating goods and chores is the tolerance of bounds used for approximating MMS values. As we discussed previously, proportionality provides a reasonable upper bound in allocating goods through reductions: as soon as the value of a bundle reaches an agent's proportionality threshold, a reduction can be applied without including any additional item.

In contrast, when allocating chores, proportionality may be a loose bound: when selecting a set of chores that satisfies proportionality for an agent, it may still be necessary to include additional chores to ensure that no chore remains unallocated.

Example 3.3. Consider an instance with 10 chores and 10 agents with identical valuations: three small chores valued at $-\frac{1}{3}$, six medium chores valued at $-\frac{1}{2}$, and one large chore valued at -1. The proportionality threshold is $-\frac{1}{2}$ but the MMS is -1. Once an agent reaches the proportionality threshold, say by receiving a single medium chore, it could still receive an additional medium or small chore.

The main challenge is how to pack as many chores as possible within a bundle without violating the maximin share threshold.

We start by making a simple assumption on the size of the instance. For any instance, without loss of generality, we can always add dummy chores with value 0 and assume that $m \geq 2d + 1$.[2]

Our first lemma will be used to bound the number of large chores in each bundle. It states that in an ordered chores instance, the most preferred $k + 1$ chores from the set of the least preferred $kd + 1$ chores are valued at least as much as 1-out-of-d MMS share.

Lemma 3.2. Let $I = \langle N, M, V\rangle$ be an ordered chores instance, and k and d be non-negative integers such that $kd + 1 \leq m$. Then, for each agent $i \in N$,

$$v_i(\{c_{kd-(k-1)}, c_{kd-(k-2)}, \ldots, c_{kd+1}\}) \geq \text{MMS}_i^d(M).$$

Proof. Consider the subset of chores $S = \{c_1, c_2, \ldots, c_{kd+1}\}$. By definition, for every chore $c \in M$, $v_i(c) \leq 0$, thus we have $\text{MMS}_i^d(S) \geq \text{MMS}_i^d(M)$. By the pigeonhole principle, since $|S| > kd$, any partition of S into d bundles $(A_1, \ldots, A_d)$ must contain at least one bundle, say A_ℓ, which contains at least $k + 1$ chores. By definition, we have $v_i(A_\ell) \geq \text{MMS}_i^d(S)$.

Let the set $B \subset S$ contain the $k + 1$ last (most preferred) chores of S. most preferred chores of S. Since chores are ordered from the least to the most preferred chores, this B is weakly preferred to A_ℓ. Thus, $v_i(A_\ell) \leq v_i(B)$ where $B = \{c_{kd-(k-1)}, c_{kd-(k-2)}, \ldots, c_{kd+1}\}$. By transitivity, $v_i(B) \geq v_i(A_\ell) \geq \text{MMS}_i^d(S)$. □

Lemma 3.2 links the number of chores to their values, and enables us to identify the number of large (least preferred) chores.

Corollary 3.1. Given an ordered chores instance $I = \langle N, M, V\rangle$, and an integer $d \geq 1$, the following statements hold[3]:

(1) $v_i(\{c\}) \geq \text{MMS}_i^d(M)$, for all $c \in M$;
(2) $v_i(\{c_d, c_{d+1}\}) \geq \text{MMS}_i^d(M)$;
(3) $v_i(\{c_{2d-1}, c_{2d}, c_{2d+1}\}) \geq \text{MMS}_i^d(M)$.

Proof. By setting $k = 0$ in Lemma 3.2, for each agent $v_i(\{c_1\}) \geq \text{MMS}_i^n(M)$. Since c_1 is the worst chore in an ordered instance, for every other chore $c \in M$, $v_i(\{c\}) \geq \text{MMS}_i^d(M)$. Similarly, setting $k = 1$ and $k = 2$ in Lemma 3.2 yields claims (2) and (3). □

4 1-OUT-OF-$\lfloor \frac{2n}{3} \rfloor$ MAXIMIN SHARE FOR CHORES IN POLYNOMIAL TIME

In this section, we present a polynomial-time algorithm for allocating chores that achieves 1-out-of-$\lfloor \frac{2n}{3} \rfloor$ MMS. The algorithm takes a chores instance along with a set of thresholds for agents as an input and utilizes a greedy "bag-filling" procedure to assign bundles of chores to agents. The high-level idea behind the algorithm is allocating the large (least desirable) chores first and packing as

[2]In the full version of the paper [38], we show this without adding dummy chores.
[3]If $d < 2d + 1$, we may add $2d + 1 - m$ dummy chores with value 0 to all agents.

ALGORITHM 1: Algorithm for 1-out-of-d MMS approximation

```
Input: An ordered chores instance $I = \langle N, M, V\rangle$ and threshold
    values $(\beta_i)_{i=1}^n$ with $\beta_i \le 0$ for all $i \in N$.
Output: Allocation $A = (A_1, \dots, A_n)$ s.t. $v_i(A_i) \ge \beta_i$ for all $i \in N$.
1 while $|N| > 0$ do // there are remaining agents
      ▷ Adding as many chores as possible to a bundle
2     Initialize $B$ as an empty bundle ;
3     for each remaining chore c in descending order of absolute values
        (hardest to easiest chore) do
4         if there exists agent $i$ s.t. $v_i(\{B \cup c\}) \ge \beta_i$ then
5             $B \leftarrow B \cup \{c\}$ // Adding c to B
      ▷ Allocating the bundle to an agent.
6     Select an agent $i$ such that $v_i(B) \ge \beta_i$ (arbitrary break ties);
7     $A_i \leftarrow B$ ;
8     $N \leftarrow N \setminus \{i\}$ ;
9     $M \leftarrow M \setminus B$ ;
```

many chores as possible into a bundle up to the given threshold. The algorithmic idea is simple. The key in achieving 1-out-of-$\lfloor \frac{2n}{3} \rfloor$ MMS approximation is selecting appropriate threshold values.

Algorithm description. The underlying structure of Algorithm 1 is similar to the *First-Fit-Decreasing* algorithm for bin-packing [40].[4] It starts by selecting an empty bundle and adding a large (lowest value) chore to the bag. While the value of the bag is above a threshold for at least one agent, add an additional chore—in order of the largest to smallest—to the bundle. If a chore cannot be added, the algorithm skips it and considers the next-smallest (more preferred) chores. Each agent has a different threshold, β_i, and assesses the bundle based on this threshold. When no more chores can be added, the bundle is allocated to an arbitrary agent who still finds it acceptable. The algorithm repeats with the remaining agents and chores.

For any selection of non-positive thresholds $(\beta_i)_{i=1}^n$, Algorithm 1 guarantees that 1) every bundle is allocated to an agent who values it at least β_i, and 2) every agent receives a bundle (possibly an empty bundle). However, if the thresholds are too optimistic (too close to zero), the algorithm may result in a partial allocation, i.e., some chores might remain unallocated. The main challenge is to carefully choose the threshold values such that the algorithm will provably terminate with a *complete* allocation.[5]

Theorem 4.1. Given an additive chores instance, a 1-out-of-$\lfloor \frac{2n}{3} \rfloor$ MMS allocation exists and can be computed in polynomial time.

Proof. Let $I = \langle N, M, V\rangle$ be an ordered instance and $d = \lfloor \frac{2n}{3} \rfloor$. Without loss of generality, we can assume that $m \ge 2d+1$ by adding dummy chores with value 0 for all agents.

For each agent, let the thresholds be selected as follows:

$$\beta_i = \min\left(v_i(c_1),\ v_i(\{c_d, c_{d+1}\}),\ v_i(\{c_{2d-1}, c_{2d},\ c_{2d+1}\}),\ \frac{v_i(M)}{d}\right).$$

Corollary 3.1 and the inequality $\frac{v_i(M)}{d} \ge \text{MMS}_i^d(M)$ imply that all agents receive their 1-out-of-d MMS, that is, $\beta_i \ge \text{MMS}_i^d(M)$.

In order to show that all chores are allocated, we split the chores into three categories of large $(\{c_1, \dots, c_d\})$, medium $(\{c_{d+1}, \dots, c_{2d}\})$, and small $(\{c_{2d+1}, \dots, c_m\})$ chores.

Since for all $i \in N$, $v_i(c_1) \ge \beta_i$, every single chore can be added to an empty bag. Consider the first d bundles. Since these bundles contain at least one chore each, and $d \le n$, the d large chores are allocated within the first d iterations.

Similarly, since $v_i(c_d, c_{d+1}) \ge \beta_i$, the medium chores may be bundled in pairs from largest to smallest and form the next bundles. This implies that, within the first $d + \lceil \frac{d}{2} \rceil$ allocated bundles, all large and medium chores are allocated. Importantly,

$$d + \left\lceil \frac{d}{2} \right\rceil = \left\lfloor \frac{2n}{3} \right\rfloor + \left\lceil \frac{\lfloor \frac{2n}{3} \rfloor}{2} \right\rceil \le \left\lfloor \frac{2n}{3} \right\rfloor + \left\lceil \frac{n}{3} \right\rceil = n.$$

Thus, we conclude that all large and medium chores are allocated upon the termination of the algorithm.

The last step is to prove that all small chores are allocated too. These chores are added to bundles whenever there is additional gap between $v_i(A_i)$ and β_i. Consider the last agent, i, who receives a bundle before Algorithm 1 terminates. If no small chores remain before agent i receives a bundle, then we are done.

Suppose that there is some remaining small chore c before agent i receives a bundle. For each other bundle A_j already allocated, necessarily $v_i(A_j \cup \{c\}) < \beta_i$, because otherwise agent i would have accepted $A_j \cup \{c\}$ and chore c would have been added to A_j. Now, since $v_i(\{c_{2d-1}, c_{2d}, c_{2d+1}\}) \ge \beta_i$ and the instance is ordered, we have that $v_i(c) \ge v_i(c_{2d+1}) \ge \frac{\beta_i}{3}$. In turn, this implies that $v_i(A_j) < \beta_i - v_i(c) = \frac{2\beta_i}{3}$ for each $j \ne i$.

By the way we selected the thresholds, we have that $\beta_i \le \frac{v_i(M)}{d}$. We use this fact to upper bound the amount of value in each previously allocated bundle:

$$v_i(A_j) < \frac{2\beta_i}{3},$$

which implies that

$$v_i(A_j) < \frac{2v_i(M)}{3d}.$$

By replacing the value of d, we have

$$v_i(A_j) < \frac{2v_i(M)}{3\lfloor \frac{2n}{3} \rfloor}.$$

Therefore,

$$v_i(A_j) \le \frac{2}{3} \cdot \frac{3}{2} \cdot \frac{v_i(M)}{n} = \frac{v_i(M)}{n}.$$

This inequality implies that before the last bundle is initialized, agent i values the remaining items at least $v_i(M) - \sum_{j \ne i} v_i(A_j) > v_i(M) - (n-1)\frac{v_i(M)}{n} = \frac{v_i(M)}{n} \ge \beta_i$. Thus, agent i can take all the remaining chores. □

Remark 4.1. Interestingly, for goods, 1-out-of-$\lfloor \frac{3n}{2} \rfloor$ MMS approximations exist [36] and can be computed in polynomial time [37]. However, the techniques used for proving the existence results as well as developing a tractable algorithm are substantially different due to reductions available for goods (as discussed in Section 3) as well as challenges posed by packing bundles to ensure complete

[4]The same algorithm is used by Huang and Lu [39] for achieving multiplicative approximations of MMS. They prove that, with appropriate thresholds, Algorithm 1 guarantees every agent at least 11/9 of its MMS value. This does not directly imply any result for ordinal approximation as shown in Example 1.1.

[5]In contrast, when allocating goods, all goods are allocated, and the challenge is showing that all *agents* receive a bundle of certain threshold.

allocations of chores. On the other hand, in the case of goods even a slight error in computing MMS values may result in wasting values and not having sufficient goods to satisfy some agents (see [36] for an example) whereas for chores we can tolerate an estimate of MMS values as long as all chores are allocated.

5 1-OUT-OF-$\lfloor \frac{3n}{4} \rfloor$ MMS ALLOCATIONS EXIST FOR CHORES

In this section, we show that a careful selection of threshold values in Algorithm 1, in fact, guarantees 1-out-of-$\lfloor \frac{3n}{4} \rfloor$ MMS approximation. To achieve this result we require a precise computation of MMS values for each agent, which in turn is intractable [16]. Nonetheless, we prove the existence of 1-out-of-$\lfloor \frac{3n}{4} \rfloor$ MMS, and later in Section 6 provide a polynomial-time algorithm that achieves an approximation of this bound.

Theorem 5.1. *Given an additive chores instance, a 1-out-of-$\lfloor \frac{3n}{4} \rfloor$ MMS allocation is guaranteed to exist.*

Theorem 5.1 is an immediate corollary of Lemma 5.1 below. For the ease of exposition, we first provide the proof of the theorem.

Proof. By construction, Algorithm 1 terminates and every agent $i \in N$ receives a bundle (possibly empty) with the value of at least β_i. By Lemma 5.1, we can pick for each agent i the threshold $\beta_i = \text{MMS}_i^d(M)$ where $d = \lfloor \frac{3n}{4} \rfloor$, and all chores will be allocated. Thus, we have a complete allocation in which each agent's value is at least 1-out-of-$\lfloor \frac{3n}{4} \rfloor$ MMS, which proves Theorem 5.1. □

Lemma 5.1. *Suppose Algorithm 1 is executed with threshold values $\beta_i \leq \text{MMS}_i^{\lfloor \frac{3n}{4} \rfloor}(M)$ for all $i \in N$. Then all chores are allocated upon termination of the algorithm.*

Proof. Let $I = \langle N, M, V \rangle$ be an ordered chores instance. For simplicity, we start by scaling the valuations such that for each agent $i \in N$, $\text{MMS}_i^{\lfloor \frac{3n}{4} \rfloor}(M) = -1$.[6] This implies that

$$v_i(M) \geq -\left\lfloor \frac{3n}{4} \right\rfloor \geq -\frac{3n}{4} \quad (1)$$

and $\beta_i \leq -1$ for each agent $i \in N$.

Let agent i be the last agent who received a bundle (in the n-th iteration). The proof proceeds by considering two types of remaining chores according to their value: 1) small chores $c \in M$ with value $v_i(c) \geq -\frac{1}{4}$, and 2) large chores $c \in M$ with value $v_i(c) < -\frac{1}{4}$.

Case 1: small chores. Suppose for contradiction that there is some chore $c \in M$ such that $v_i(c) \geq -\frac{1}{4}$ that remains unallocated at the end of the algorithm. By assumption, agent i could not add c to any allocated bundle, including i's own bundle. Since i is the last agent, we infer that for each agent $j \in N$ with bundle A_j, $v_i(A_j \cup \{c\}) < -1$. By additivity, because $v_i(c) \geq -\frac{1}{4}$, we can write $v_i(A_j) < -\frac{3}{4}$ for all $j \in N$. Summing over all assigned bundles gives $v_i(M) < -\frac{3n}{4}$, which contradicts (1). Therefore, no such small chore remains at the end of the algorithm.

[6]This scaling step is only used to simplify the proof. An identical result can be achieved without scaling the valuations by setting all thresholds to $\beta_i = \text{MMS}_i^d(M)$ where $d = \lfloor \frac{3n}{4} \rfloor$ and updating the rest of the values in the proof accordingly.

Case 2: large chores. Suppose that there is some chore $c \in M$ such that $v_i(c) < -\frac{1}{4}$ that remains unallocated at the end of the algorithm. We define the following sets of bundles.

- $M_1, \ldots, M_{\lfloor \frac{3n}{4} \rfloor}$ are *MMS bundles* — bundles that comprise a $\text{MMS}_i^{\lfloor \frac{3n}{4} \rfloor}(M)$ partition of agent i.
- $B_1, \ldots, B_n$ are *algorithm bundles* — bundles allocated by Algorithm 1. B_t denotes the bundle allocated at iteration t.

For each MMS bundle M_j, let $M_j[s]$ denote the s-th largest chore (least valued) of M_j. Whenever $|M_j| < s$, we define $v_i(M_j[s]) = 0$. Without loss of generality, we assume that the MMS bundles are sorted such that $|v_i(M_1[1])| \geq |v_i(M_2[1])| \geq \ldots \geq |v_i(M_{\lfloor \frac{3n}{4} \rfloor}[1])|$. Since valuations are scaled so that $\text{MMS}_i^{\lfloor \frac{3n}{4} \rfloor}(M) = -1$, there are at most 3 large chores (with value less than $\frac{1}{4}$) in each MMS bundle.

For the sake of the proof, we maintain a vector of *shadow-bundles* $M'_1, M'_2, \ldots, M'_n$, which is initialized as follows:

- For each $j \in \{1, \ldots, \lfloor \frac{3n}{4} \rfloor\}$, $M'_j :=$ the set of large chores (with value less than $-\frac{1}{4}$ to i) in M_j.
- For each $j \in \{\lfloor \frac{3n}{4} \rfloor, \ldots, n\}$, $M'_j := \emptyset$.

At each iteration t of the algorithm, we edit the vector of shadow-bundles by moving some chores between bundles. We do so such that, at the start of iteration t, the following invariants hold:

(1) $M'_j \subseteq B_j$ for all $j < t$. That is, each chore in the shadow-bundles $M'_1, \ldots, M'_{t-1}$ is allocated.
(2) $|M'_j| \leq 3$ and $v_i(M'_j) \geq -1$ for $j \geq t$. That is, each remaining shadow-bundle $M'_t, \ldots, M'_n$ has value at least -1.

Both invariants hold before the first iteration ($t = 1$): invariant (1) holds vacuously, and invariant (2) holds since each bundle M'_j is contained in one of i's MMS bundles. Suppose the invariants hold before iteration $t \geq 1$. We show how to edit the shadow-bundles such that the invariants still hold before iteration $t + 1$.

We reorder the shadow-bundles $M'_t, \ldots, M'_n$ so that $M'_t[1]$ is the largest remaining chore. Hence, in iteration t, Algorithm 1 selects this chore first to add to the bag. That is, $B_t[1] = M'_t[1]$. We split to cases based on the size of $|M'_t|$, which must be in $\{1, 2, 3\}$ by invariant (2).

If $|M'_t| = 1$, then both invariants hold at $t+1$, since $M'_t \subseteq B_t$, and the shadow-bundles do not change.

If $|M'_t| = 2$, then we have to handle $M'_t[2]$. By invariant (2) we have $M'_t[1] + M'_t[2] \geq -1$. This means that $M'_t[2]$ can potentially be inserted as the second chore in B_t. If indeed $B_t[2] = M'_t[2]$, then we are done — both invariants hold at $t+1$, since $M'_t \subseteq B_t$, and the shadow-bundles do not change. If $B_t[2] \neq M'_t[2]$, this means that Algorithm 1 processed chore $B_t[2]$ before chore $M'_t[2]$. Since the algorithm processes jobs by ascending order of values (descending order of absolute values), this implies that $v_i(B_t[2]) \leq v_i(M'_t[2])$. Now, we find the chore $B_t[2]$ in some shadow-bundle M'_j for some $j > t$, and swap it with $M'_t[2]$. We claim that both invariants hold:

(1) $M'_t \subseteq B_t$, since after the swap $B_t[1] = M'_t[1]$ and $B_t[2] = M'_t[2]$, and $|M'_t| = 2$.
(2) The remaining shadow bundles remained as before, except for the shadow-bundle M'_j, in which a single chore was swapped. But, because $v_i(B_t[2]) \leq v_i(M'_t[2])$, the value of M'_j weakly increases, so it is still at least -1.

Finally, suppose $|M'_t| = 3$. We handle $M'_t[2]$ as in the previous case, so that now $M'_t[1] = B_t[1]$ and $M'_t[2] = B_t[2]$. It remains to handle $M'_t[3]$. Because $M'_t[3]$ is the smallest chore in M'_t, and $v_i(M'_t) \geq -1$, by the pigeonhole principle we must have $v_i(M'_t[3]) \geq -\frac{1}{3}$. We move chore $M'_t[3]$ to a bundle M'_j which was initially empty and which contains fewer than 3 chores (all of which were moved to the bundle this way and thus have value at least $-\frac{1}{3}$). Such a bundle can always be found because at most one chore is moved this way in each iteration, and there are at most $\lfloor \frac{3n}{4} \rfloor$ bundles M'_j which were initially non-empty. Thus an upper bound on the number of bundles filled this way is: $\lceil \frac{1}{3} \cdot \lfloor \frac{3n}{4} \rfloor \rceil \leq \frac{n}{4} \leq n - \lfloor \frac{3n}{4} \rfloor$. Since each chore moved this way has value at least $-\frac{1}{3}$, we preserve invariant (2) $|M'_j| \leq 3$ and $v_i(M'_j) \geq -1$. After the move, M'_t contains only two chores, both of which are in B_t, so invariant (1) holds too.

We note that if the first chore $B_t[1]$ is selected from one of these growing bundles, then because this chore has value at least $-\frac{1}{3}$ and because chores are only moved if $v_i(M'_t[1]) < -\frac{1}{3}$, no more chores will be moved in later iterations.

The final step in proving the lemma is to move all chores from $B_t \setminus M'_t$ to M'_t. This step is necessary in order to guarantee that the largest remaining chore in later steps is not from $B_t \setminus M'_t$ (and thus $M'_{t+1}[1] \notin B_t \setminus M'_t$).[7] We may do this because it preserves $M'_t \subseteq B_t$. Notice that $v_j(B_t) \geq -1$ for the agent $j \in N$ who received bundle B_t; however, we do not require that $v_i(B_t) \geq -1$, as agent i is not be allocated the bundle B_t. Observe that the chores $B_t \setminus M'_t$ correspond to additional large chores which could be added to the bundle M'_t, and thus, in moving these chores, the value of bundles M'_j for $j > t$ can only weakly increase and will remain at least -1.

Lastly, invariant (2) implies that after iteration $(n-1)$, M'_n has value at least -1 for agent i. All remaining large chores lie in this bundle. Thus agent i may take all such large chores. This implies that $M'_n \subseteq B_n$ and that no large chores remain when the algorithm terminates. □

We do not know whether the $\lfloor 3n/4 \rfloor$ factor is tight in general. The following proposition shows a non-tight upper bound on the performance of Algorithm 1 for large values of n.

Proposition 5.1 (Upper bound for Algorithm 1). For any integer $k \geq 0$, there is an instance with $n = 11k + 7$ agents in which Algorithm 1 cannot guarantee to each agent its 1-out-of-$(9k + 6)$ MMS.

Proof. When all agents have the same valuation and the same threshold, Algorithm 1 reduces to an algorithm for bin-packing known as *First Fit Decreasing* (*FFD*) [12, 40]. FFD sorts the chores by descending value, and allocates each chore to the first (smallest-index) agent who can take it without going over the threshold. Algorithm 1 (with identical valuations and thresholds) does exactly the same, only in a different order: instead of making a single pass over all the chores and filling all bins simultaneously, it makes n passes over the chores, and fills each bin in turn with the chores that would be inserted to it in that single pass.

[7]For example, consider $M'_1 = \{c_1, c_2\}$ and $M'_2 = \{c_3, c_4, c_5\}$. It is possible that $B_1 = \{c_1, c_2, c_3\}$, which means that $B_2[1] = c_4$ but $M'_2[1] = c_3$.

Dósa [24] and Dósa et al. [25] have shown that, for every integer $k \geq 1$, there is a bin packing instance in which the optimal packing needs $9k + 6$ bins but FFD needs $11k + 8$ bins. We construct a chore allocation instance with $n = 11k + 7$ agents with identical valuations, taken from that bin-packing instance. Assume that the agents' thresholds are at least their 1-out-of-$(9k + 6)$ MMS. Then, after Algorithm 1 allocates bundles to all n agents, some chores may remain unallocated. □

Consider Proposition 5.1 with $k = 0$ and $n = 7$. By Theorem 5.1, our algorithm achieves $\lfloor 3n/4 \rfloor = 5$ ordinal approximation. This bound is tight since we cannot guarantee to all agents their 1-out-of-6 MMS. We present this tight example below.

Example 5.1 (A tight example for Algorithm 1)**.** Consider an instance with $n = 7$ agents and $m = 20$ chores valued as follows for all agents: four chores valued at -201, four chores valued at -102, four chores valued at -101, and eight chores valued at -98. For each agent, the 1-out-of-6 MMS partition contains the following bundles with the MMS value of -400:

- 4 bundles of chores with values $\{-201, -101, -98\}$;
- 2 bundles of chores with values $\{-102, -102, -98, -98\}$.

With the threshold values set as -400, Algorithm 1 generates the following bundles:

- 4 bundles with chores $\{-201, -102\}$;
- 1 bundle with chores $\{-101, -101, -101\}$;
- 1 bundle with chores $\{-101, -98, -98, -98\}$;
- 1 bundle with chores $\{-98, -98, -98, -98\}$.

After allocating these 7 bundles, a chore with the value of -98 remains unallocated and cannot be added to any of the above bundles since it would violate the threshold of -400.

6 POLYNOMIAL-TIME APPROXIMATIONS

In this section, we develop an efficient approximation algorithm that achieves 1-out-of-$\lfloor \lfloor \frac{3n}{4} \rfloor - O(\log n) \rfloor$ MMS for any chores instance. We rely on Algorithm 1 while utilizing an efficient approximation algorithm to find reasonable threshold values.

This result provides an interesting computational contrast between multiplicative and ordinal approximations of MMS for allocating chores: multiplicative approximations require exact MMS values, which can be seen as a *job scheduling* problem where the goal is to minimize the makespan (the maximum completion time of a machine). However, ordinal MMS approximation on chore instances can be modeled as a combinatorial problem of *bin packing* (see Korte and Vygen [41] for a detailed survey) where the goal is to minimize the number of bins subject to an upper bound on the total size of items in each bin.

While both problems are NP-hard, they differ in the approximation algorithms available for them. The job scheduling problem has polynomial-time approximation schemes (PTAS) [54], but their runtime is exponential in the approximation accuracy $1/\epsilon$. On the other hand, the bin packing problem used for our ordinal MMS approximation admits *additive* approximation algorithms.

In particular, we use an algorithm by Hoberg and Rothvoss [34], which we call *Algorithm HR.* Algorithm HR takes as input a bin-packing instance I, and returns a packing with at most $\lceil OPT(I) + a \cdot \log(OPT(I)) \rceil$ bins (for some fixed constant a) in time polynomial

ALGORITHM 2: Computing an approximate MMS value

```
Input: An integer $d \geq 1$; a single agent with value function $v_i$ over
    a set of chores $M$; all values are negative integers.
Output: A number $\beta_i$ in the interval
    $[\mathrm{MMS}_i^{\lfloor d - \log d \rfloor}(M), \mathrm{MMS}_i^{d}(M)]$.
▷ Construct a bin-packing instance:
1  Let $S := \{ -v_i(c) \mid c \in M \}$ ;
▷ Initialize a lower and an upper bound for the bin
  size:
2  Let $L := 0$ ;
3  Let $U := (\sum S)$ rounded up to the nearest power of 2;
▷ Run binary search:
4  while $U > L + 1$ do
5      Let $b := (U + L)/2$ ;
6      Run Algorithm HR [34] on instance $S$ with bin-size $b$ ;
7      if at most $d$ bins are used then
8          Let $U := b$;        // Try smaller bins
9      else
10         Let $L := b$;        // Try larger bins
11 return $-U$.
```

in m (the number of input numbers in I), where $OPT(I)$ denotes the smallest possible number of bins for I. We combine Algorithm HR with binary search on the bin size.[8]

To efficiently apply binary search, we assume in this section that the values of chores are negative *integers* with a bounded binary representation. The run-time of our algorithm will be polynomial in the size of the binary representation of the input.

Lemma 6.1. Given an additive chores instance with integer values, for any integer $d \geq 1$ and agent i, it is possible to compute a number β_i for which

$$\mathrm{MMS}_i^{\lfloor d - a \cdot \log d \rfloor}(M) \leq \beta_i \leq \mathrm{MMS}_i^{d}(M) \leq 0,$$

in time polynomial in the size of binary representation of the input.

Proof. We start by applying Algorithm 2. The algorithm converts the chores allocation instance to a bin-packing instance, where each chore $c \in M$ is converted to an input of size $|v_i(c)|$. Then it applies binary search with lower bound L and upper bound U. Throughout the search, the following invariants are maintained:

(1) $U > L \geq 0$;
(2) Algorithm HR with bin-size U needs at most d bins;
(3) Algorithm HR with bin-size L needs more than d bins.

The invariants are obviously true at initialization, and they are maintained by the way U and L are updated. Let β_i be the returned value, that is, the value of $-U$ once the algorithm terminates. By the termination condition, at this point $U = L + 1$.

Invariant (2) implies that there exists a partition of chores into d bins, in which the total absolute value of each bin is at most U, so the total value is at least $-U$. Therefore, $\mathrm{MMS}_i^{d}(M) \geq -U = \beta_i$.

Invariant (3) implies that there is no partition of the chores into $\lfloor d - a \cdot \log d \rfloor$ or fewer bins, in which the total absolute value of all bins is at most L—otherwise the HR algorithm could have filled at most $\lceil \lfloor d - a \cdot \log d \rfloor + a \cdot \log \lfloor d - a \cdot \log d \rfloor \rceil \leq d$ bins of size L. Therefore, $\mathrm{MMS}_i^{\lfloor d - a \cdot \log d \rfloor}(M) < -L$. Since we assumed that all chores' values are integers, this implies $\mathrm{MMS}_i^{\lfloor d - a \cdot \log d \rfloor}(M) \leq -L - 1 = -U = \beta_i$.

The binary search uses $\lceil \log_2(\sum S) \rceil$ iterations, which is polynomial in the size of the binary representation of the input. Each iteration runs the HR algorithm, whose run-time is polynomial in m. This concludes the proof of the lemma. □

[8]Similar search techniques have been used for *MultiFit* scheduling algorithms [23] and the dual approximation scheme of Hochbaum and Shmoys [35].

ALGORITHM 3: Algorithm for ordinal MMS approximation in polynomial time

```
Input: An ordered chores instance $I = \langle N, M, V \rangle$.
Output: Allocation $A = (A_1, \ldots, A_n)$ satisfying
    $v_i(A_i) \geq \mathrm{MMS}_i^{d}(M)$ for all $i \in N$, such that
    $d = \lfloor \lfloor \frac{3n}{4} \rfloor - a \cdot \log \lfloor \frac{3n}{4} \rfloor \rfloor$
1  for each agent $i \in N$: do
2      Run Algorithm 2 with $d = \lfloor \frac{3n}{4} \rfloor$ and valuation $v_i$;
3      Let $\beta_i$ be the returned value;
4  Run Algorithm 1 on $I$ with the threshold values $(\beta_1, \ldots, \beta_n)$.
```

Theorem 6.1. Given an additive chores instance with integer values, it is possible to find in polynomial time, for some fixed positive constant a, a 1-out-of-$\lfloor \lfloor \frac{3n}{4} \rfloor - a \cdot \log \lfloor \frac{3n}{4} \rfloor \rfloor$ MMS allocation

Proof. We use Algorithm 3 which starts by computing a threshold value β_i for each agent $i \in N$ using Algorithm 2. Then, it applies Algorithm 1 with the resulting thresholds for allocating the chores.

Lemma 6.1 implies that $\beta_i \leq \mathrm{MMS}_i^{d}(M)$ with $d = \lfloor \frac{3n}{4} \rfloor$ for all $i \in N$. By Lemma 5.1, this implies that Algorithm 1 allocates all the chores. Therefore, Algorithm 1 yields a complete allocation in which the value of each agent i is at least β_i. By Lemma 6.1, this value is at least $\mathrm{MMS}_i^{\lfloor \lfloor \frac{3n}{4} \rfloor - a \cdot \log \lfloor \frac{3n}{4} \rfloor \rfloor}(M)$, concluding the proof. □

7 DISCUSSION

Theorem 5.1 shows that, asymptotically (when n is large), Algorithm 1 guarantees 1-out-of-($\approx 0.75n$) MMS. Proposition 5.1, however, shows that this bound cannot be improved to 1-out-of-($\approx 0.81n$) MMS using this algorithm. An immediate, but challenging, research direction is closing this approximation gap and developing polynomial-time algorithms beyond those presented in this paper.

All of our results use the same algorithm (Algorithm 1) to allocate the chores, but with different threshold values. This approach is "pluralistic" in that it allows each agent to choose between these thresholds: each agent may choose whether to settle for a lower but easy-to-compute threshold of Section 4, or put an extra effort to compute a higher thresholds of Sections 5 or 6. This pluralistic approach may be useful in other fair division settings.

ACKNOWLEDGMENTS

Hadi Hosseini acknowledges support from NSF IIS grants #2052488 and #2107173. Erel Segal-Halevi is supported by the ISF grant 712/20. We are grateful to anonymous referees of EC 2021 and AAMAS 2022 for their valuable feedback.

REFERENCES

[1] Elad Aigner-Horev and Erel Segal-Halevi. 2022. Envy-free matchings in bipartite graphs and their applications to fair division. *Information Sciences* 587 (2022), 164–187.

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

[3] Haris Aziz, Ioannis Caragiannis, Ayumi Igarashi, and Toby Walsh. 2019. Fair allocation of combinations of indivisible goods and chores. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*. 53–59.

[4] Haris Aziz, Hau Chan, and Bo Li. 2019. Maxmin share fair allocation of indivisible chores to asymmetric agents. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 1787–1789.

[5] Haris Aziz, Hau Chan, and Bo Li. 2019. Weighted Maxmin Fair Share Allocation of Indivisible Chores. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 46–52. https://doi.org/10.24963/ijcai.2019/7

[6] Haris Aziz, Bo Li, and Xiaowei Wu. 2019. Strategyproof and Approximately Maxmin Fair Share Allocation of Chores. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 60–66. https://doi.org/10.24963/ijcai.2019/9

[7] Haris Aziz, Bo Li, and Xiaowei Wu. 2020. Approximate and Strategyproof Maximin Share Allocation of Chores with Ordinal Preferences. *arXiv preprint arXiv:2012.13884* (2020).

[8] Haris Aziz, Hervé Moulin, and Fedor Sandomirskiy. 2020. A polynomial-time algorithm for computing a Pareto optimal and almost proportional allocation. *Operations Research Letters* 48, 5 (2020), 573–578.

[9] Haris Aziz, Gerhard Rauchecker, Guido Schryen, and Toby Walsh. 2017. Algorithms for Max-Min Share Fair Allocation of Indivisible Chores. *Proceedings of the AAAI Conference on Artificial Intelligence* 31, 1 (Feb. 2017). https://ojs.aaai.org/index.php/AAAI/article/view/10582

[10] Moshe Babaioff, Noam Nisan, and Inbal Talgam-Cohen. 2019. Fair Allocation through Competitive Equilibrium from Generic Incomes. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 180–180.

[11] Moshe Babaioff, Noam Nisan, and Inbal Talgam-Cohen. 2021. Competitive equilibrium with indivisible goods and generic budgets. *Mathematics of Operations Research* 46, 1 (2021), 382–403.

[12] Brenda S Baker. 1985. A new proof for the first-fit decreasing bin-packing algorithm. *Journal of Algorithms* 6, 1 (1985), 49–70.

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

[14] Anna Bogomolnaia and Hervé Moulin. 2022. Guarantees in Fair Division: General or monotone preferences. arXiv preprint 1911.10009.

[15] Anna Bogomolnaia, Hervé Moulin, Fedor Sandomirskiy, and Elena Yanovskaia. 2019. Dividing bads under additive utilities. *Social Choice and Welfare* 52, 3 (2019), 395–417.

[16] Sylvain Bouveret and Michel Lemaître. 2016. Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Autonomous Agents and Multi-Agent Systems* 30, 2 (01 Mar 2016), 259–290. https://doi.org/10.1007/s10458-015-9287-3

[17] Steven J Brams and Alan D Taylor. 1996. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press.

[18] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. 2016. *Handbook of computational social choice*. Cambridge University Press.

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

[20] Simon Caney. 2009. Justice and the distribution of greenhouse gas emissions. *Journal of global ethics* 5, 2 (2009), 125–146.

[21] Bhaskar Ray Chaudhury, Jugal Garg, Peter McGlaughlin, and Ruta Mehta. 2020. Dividing bads is harder than dividing goods: On the complexity of fair and efficient division of chores. *arXiv preprint arXiv:2008.00285* (2020).

[22] Xingyu Chen and Zijie Liu. 2020. The fairness of leximin in allocation of indivisible chores. *arXiv preprint arXiv:2005.04864* (2020).

[23] Edward G Coffman, Jr, Michael R Garey, and David S Johnson. 1978. An application of bin-packing to multiprocessor scheduling. *SIAM J. Comput.* 7, 1 (1978), 1–17.

[24] György Dósa. 2007. The tight bound of first fit decreasing bin-packing algorithm is FFD (I) <= 11/9OPT (I)+ 6/9. In *International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Springer, 1–11.

[25] György Dósa, Rongheng Li, Xin Han, and Zsolt Tuza. 2013. Tight absolute bound for First Fit Decreasing bin-packing: FFD (L) <= 11/9 OPT (L)+ 6/9. *Theoretical Computer Science* 510 (2013), 13–61.

[26] Edith Elkind, Erel Segal-Halevi, and Warut Suksompong. 2021. Graphical Cake Cutting via Maximin Share. In *Proceedings of IJCAI*.

[27] Edith Elkind, Erel Segal-Halevi, and Warut Suksompong. 2021. Mind the Gap: Cake Cutting With Separation. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 5330–5338.

[28] Uriel Feige, Ariel Sapir, and Laliv Tauber. 2021. A tight negative example for MMS fair allocations. *arXiv preprint arXiv:2104.04977* (2021).

[29] Rupert Freeman, Sujoy Sikdar, Rohit Vaish, and Lirong Xia. 2020. Equitable Allocations of Indivisible Chores. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 384–392.

[30] Jugal Garg, Peter McGlaughlin, and Setareh Taki. 2018. Approximating Maximin Share Allocations. In *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

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

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

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

[34] Rebecca Hoberg and Thomas Rothvoss. 2017. A logarithmic additive integrality gap for bin packing. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2616–2625.

[35] Dorit S Hochbaum and David B Shmoys. 1987. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)* 34, 1 (1987), 144–162.

[36] Hadi Hosseini and Andrew Searns. 2021. Guaranteeing Maximin Shares: Some Agents Left Behind. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Zhi-Hua Zhou (Ed.). International Joint Conferences on Artificial Intelligence Organization, 238–244. https://doi.org/10.24963/ijcai.2021/34 Main Track.

[37] Hadi Hosseini, Andrew Searns, and Erel Segal-Halevi. 2021. Ordinal Maximin Share Approximation for Goods. *arXiv preprint arXiv:2109.01925* (2021).

[38] Hadi Hosseini, Andrew Searns, and Erel Segal-Halevi. 2022. Ordinal Maximin Share Approximation for Chores. arXiv:2201.07424 [cs.GT]

[39] Xin Huang and Pinyan Lu. 2021. An algorithmic framework for approximating maximin share allocation of chores. In *Proceedings of the 22nd ACM Conference on Economics and Computation (EC)*. 630–631.

[40] David S Johnson. 1973. *Near-optimal bin packing algorithms*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[41] Bernhard Korte and Jens Vygen. 2018. Bin-Packing. In *Combinatorial Optimization*. Springer, 489–507.

[42] Rucha Kulkarni, Ruta Mehta, and Setareh Taki. 2021. Indivisible Mixed Manna: On the Computability of MMS + PO Allocations. In *Proceedings of the 22nd ACM Conference on Economics and Computation*. 683–684.

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

[44] Peter McGlaughlin and Jugal Garg. 2020. Improving Nash social welfare approximations. *Journal of Artificial Intelligence Research* 68 (2020), 225–245.

[45] Hervé Moulin. 2019. Fair Division in the Internet Age. *Annual Review of Economics* 11, 1 (2019), 407–441. https://doi.org/10.1146/annurev-economics-080218-025559 arXiv:https://doi.org/10.1146/annurev-economics-080218-025559

[46] Nhan-Tam Nguyen, Trung Thanh Nguyen, and Jörg Rothe. 2017. 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*. International Foundation for Autonomous Agents and Multiagent Systems, 262–271.

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

[48] Mathias Risse. 2008. *Who Should Shoulder the Burden? Global Climate Change and Common Ownership of the Earth*. Technical Report. Harvard University, John F. Kennedy School of Government.

[49] Andrew Searns and Hadi Hosseini. 2020. Fairness Does Not Imply Satisfaction (Student Abstract). *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 10 (Apr. 2020), 13911–13912. https://doi.org/10.1609/aaai.v34i10.7228

[50] Erel Segal-Halevi. 2019. The Maximin Share Dominance Relation. arXiv preprint 1912.08763.

[51] Erel Segal-Halevi. 2020. Competitive equilibrium for almost all incomes: existence and fairness. *Autonomous Agents and Multi-Agent Systems* 34, 1 (2020), 1–50.

[52] Martino Traxler. 2002. Fair chore division for climate change. *Social Theory and Practice* 28, 1 (2002), 101–134.

[53] Miroslaw Truszczynski and Zbigniew Lonc. 2020. Maximin Share Allocations on Cycles. *Journal of Artificial Intelligence Research* 69 (2020), 613–655.

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