Improved Maximin Share Approximations for Chores by Bin Packing

Jugal Garg[1], **Xin Huang**[2], **Erel Segal-Halevi**[3]
[1]University of Illinois at Urbana Champaign, USA
[2]Kyushu University, Fukuoka, Japan
[3]Ariel University, Ariel 40700, Israel
jugal@illinois.edu, huangxin@inf.kyushu-u.ac.jp, erelsgl@gmail.com

Abstract

We study fair division of indivisible chores among n agents with additive cost functions using the popular fairness notion of maximin share (MMS). Since MMS allocations do not always exist for more than two agents, the goal has been to improve its approximations and identify interesting special cases where MMS allocations exists. We show the existence of

- 1-out-of-$\lfloor \frac{9}{11} n \rfloor$ MMS allocations, which improves the state-of-the-art factor of 1-out-of-$\lfloor \frac{3}{4} n \rfloor$.
- MMS allocations for factored instances, which resolves an open question posed by Ebadian et al. (2021).
- $15/13$-MMS allocations for personalized bivalued instances, improving the state-of-the-art factor of $13/11$.

We achieve these results by leveraging the HFFD algorithm of Huang and Lu (2021). Our approach also provides polynomial-time algorithms for computing an MMS allocation for factored instances and a $15/13$-MMS allocation for personalized bivalued instances.

Introduction

Fair division of indivisible tasks (or chores) has garnered significant attention recently due to its applications in various multi-agent settings; see recent surveys (Amanatidis et al. 2023; Liu et al. 2024). The problem is to find a *fair* partition of a set $\mathcal{M}$ of m indivisible chores among n agents with preferences. We assume that each agent i has additive preferences represented by the cost functions $v_i(.)$ such that the cost of a set of chores S is given by $v_i(S) = \sum_{c \in S} v_i(c)$, where $v_i(c)$ represents the cost of chore c for agent i.

A natural and popular fairness notion in the context of indivisible items is called maximin share (MMS), introduced by Budish (2011). It appears to be also favored by participating agents in real-life experiments (Gates, Griffiths, and Dragan 2020). An agent's MMS is defined as the minimum cost they can ensure by partitioning all the chores into n bundles (one for each agent) and then receiving a bundle with the highest cost. Formally, for a set S of chores and an integer d, let $\Pi_d(S)$ denote the set of all partitions of S into d bundles. Then,

$$\text{MMS}_i^d(S) := \min_{(S_1,\ldots,S_d)\in\Pi_d(S)} \max_j v_i(S_j).$$

Let us denote the MMS of an agent i by $\mu_i := \text{MMS}_i^n(\mathcal{M})$. In an MMS allocation, each agent's bundle cost does not exceed their MMS. However, for more than two agents, MMS allocations are not guaranteed to exist (Aziz et al. 2017; Feige, Sapir, and Tauber 2021). Therefore, the focus shifted to exploring approximations of MMS and identifying interesting special classes where MMS allocations can be achieved. Two natural relaxations are multiplicative and ordinal approximations.

α-MMS This approach involves multiplying the MMS by a factor $\alpha > 1$ to raise each agent's threshold. An allocation is said to be α-MMS if the cost of each agent's bundle is at most α times their MMS. Research has progressed to demonstrate the existence of $13/11$-MMS allocations (Huang and Segal-Halevi 2023).

1-out-of-d-MMS Another way to adjust the threshold is by considering the MMS when the chores are divided into $d < n$ bundles. An allocation is 1-out-of-d-MMS if each agent's bundle cost is no more than $\text{MMS}_i^d(\mathcal{M})$. This relaxation, initially introduced in (Budish 2011) for the case of goods, is valued for its resilience to small perturbations in chores costs; see (Hosseini, Searns, and Segal-Halevi 2021) for more details. The current best-known factor for which existence is established is 1-out-of-$\lfloor \frac{3}{4} n \rfloor$ (Hosseini, Searns, and Segal-Halevi 2022).

Our Contributions

In this paper, we advance the state-of-the-art on all three fronts: achieving exact MMS, and exploring both multiplicative and ordinal approximations. We show the existence of

- **1-out-of-$\lfloor \frac{9}{11} n \rfloor$ MMS allocations** for all additive instances, improving the current best-known factor of 1-out-of-$\lfloor \frac{3}{4} n \rfloor$.
- **MMS allocations for factored instances**, where each $v_i(c) \in \{p_1, p_2, \ldots, p_k\}$ such that $p_\ell = p_{\ell-1} \cdot q$ for some integer $q > 0$ for each $\ell \in [k-1]$. Factored instances encompass the well-studied class of weakly lexicographic preferences (Aziz et al. 2019; Hosseini et al. 2021). This contribution also resolves an open question posed by (Ebadian, Peters, and Shah 2021).

- $15/13$**-MMS allocations for personalized bivalued instances**, where each $v_i(c) \in \{a_i, b_i\}$ for some positive rational numbers a_i, b_i. They generalize the well-studied bivalued instances, where each $v_i(c) \in \{a, b\}$ for some positive constants a, b, as the values of a_i, b_i can vary between different agents. This is better than the factor of $13/11$ for general instances.

We achieve these results by leveraging the Heterogeneous First Fit Decreasing (HFFD) algorithm of Huang and Lu (2021). The HFFD algorithm is a heterogeneous variant of the classic First Fit Decreasing (FFD) algorithm used in the bin packing problem (Johnson 1973), with an approximation factor proven to be $13/11$ (Huang and Segal-Halevi 2023). As in previous works on MMS, the algorithm is straightforward; however, the novelty and challenge lie in the intricate analysis.

For our first result, we extend the analysis of HFFD algorithm in (Huang and Segal-Halevi 2023) to the ordinal approximation of MMS, providing an improved bound. Our second result demonstrates that the HFFD algorithm is optimal for factored instances. Our third result establishes that the HFFD algorithm attains a factor of $15/13$ for personalized bivalued instances through a detailed case analysis.

Additionally, our approach results in polynomial-time algorithms for the second and third contributions, enabling the computation of an MMS allocation for factored instances and a $15/13$-MMS allocation for personalized bivalued instances in polynomial time.

Technically, our proofs adopts a consistent analysis framework. We begin with a base allocation, which is either an MMS allocation or the output of the FFD algorithm. While the exact structure of this base allocation is unknown, its existence is guaranteed. We conceptualize the allocation produced by the HFFD algorithm as being constructed through an iterative process. This process starts with the base allocation and incrementally modifies it by swapping chores between bundles, transforming the k-th bundle of the current allocation into the k-th bundle output by the HFFD algorithm. Using this iterative framework, we employ induction to show that the final allocation satisfies the desired properties. The key to understanding our proof lies in visualizing the swapping process and ensuring that each swap does not increase the cost of subsequent bundles.

Further Related Work

Given the intense study of MMS notion and its variants, we focus on closely related work here.

Computing the MMS value of an agent is NP-hard, even for two agents, using a straightforward reduction from the Partition problem. However, a Polynomial Time Approximation Scheme (PTAS) exists for this computation using a PTAS for the job scheduling problem (Hochbaum and Shmoys 1987). However, for factored instances, MMS values can be computed in polynomial time (Ebadian, Peters, and Shah 2021).

MMS allocations are not guaranteed to exist for more than two agents (Aziz et al. 2017; Feige, Sapir, and Tauber 2021), which has motivated the exploration of approximate MMS allocations to ensure their existence. For multiplicative approximation, a series of works (Aziz et al. 2017; Barman and Krishnamurthy 2020; Huang and Lu 2021; Huang and Segal-Halevi 2023) have established the current best approximation factor of $13/11$. On the other hand, for ordinal approximation, research has progressed to show the existence of 1-out-of-$\lfloor \frac{3}{4}n \rfloor$ MMS allocations (Aigner-Horev and Segal-Halevi 2022; Hosseini, Searns, and Segal-Halevi 2022). For the special case of (non-personalized) bivalued instances, MMS allocations are known to exist (Feige 2022).

Goods The MMS notion can similarly be defined for the fair division of goods. Like the case of chores, MMS allocations for goods do not always exist (Kurokawa, Procaccia, and Wang 2018). Extensive research has been dedicated to approximate MMS allocations for goods. Notable works (Amanatidis et al. 2017; Ghodsi et al. 2018; Garg and Taki 2021; Akrami et al. 2023; Hosseini and Searns 2021; Hosseini, Searns, and Segal-Halevi 2021) have led to a multiplicative approximation factor of $3/4 + 3/3836$ (Akrami and Garg 2024) and an ordinal approximation factor of 1-out-of-$4\lfloor n/3 \rfloor$ (Akrami et al. 2024).

Preliminaries

We denote $[t] = \{1, \ldots, t\}$. An instance of the problem of dividing a set $\mathcal{M}$ of m indivisible items (chores) among n agents is given by a cost function vector $v = (v_1, \ldots, v_n)$, where v_i denotes the additive cost function of agent i. We assume that all chore costs are positive, i.e., $v_i(c) > 0$ for all $c \in \mathcal{M}$.

IDO instances It is without loss of generality to assume that the instance has identical order preferences (IDO), meaning there exists a universal ordering of all chores $c_1, \ldots, c_m$ such that, for any agent i, $v_i(c_1) \geq v_i(c_2) \cdots \geq v_i(c_m)$ (Huang and Lu 2021). Thus, we will assume a *universal ordering* over the chores, which is also used for tie-breaking.

FFD algorithm The chores division problem has a strong connection with the well-studied bin packing problem. Huang and Lu (2021) were the first to apply the First Fit Decreasing (FFD) algorithm, a technique from the bin packing problem, to the fair division of chores, a method further utilized in Huang and Segal-Halevi (2023).

Given a set $\mathcal{M}$ of chores, a cost function v on $\mathcal{M}$, and a threshold τ (the "bin size"), the FFD algorithm first orders the chores by descending cost, based on the cost function v. It then packs each chore in the order into the first available bin where it fits, meaning the total cost in the bin does not exceed the threshold τ. If no existing bin can accommodate the chore without surpassing the threshold, a new empty bin is opened, and the chore is placed in it. We denote the sequence of bins output by the FFD algorithm for a given set of inputs as $FFD(\mathcal{M}, v, \tau)$.

MultiFit algorithm To ensure that the number of bins exactly equals n (one for each agent), we employ the MultiFit algorithm (Coffman, Garey, and Johnson 1978). This algorithm runs the FFD algorithm multiple times with different threshold values τ. The goal is to find a τ that allows

$FFD(\mathcal{M}, v, \tau)$ to allocate all the chores into exactly n bins. It uses binary search to efficiently determine the threshold.

HFFD algorithm Huang and Lu (2021) extended the FFD algorithm from bins to agents to accommodate the scenario where agents have distinct cost functions v_i's and different thresholds τ_i's. This generalized algorithm is known as Heterogeneous FFD (HFFD). The HFFD algorithm processes chores by the universal ordering (from largest to smallest cost). It proceeds by filling one bin at a time as follows: for each chore, the algorithm checks if the chore fits into the current bin according to the threshold τ_i and the cost function v_i of at least one remaining agent i. If it does fit, the chore is added to the bin; otherwise, it is skipped. Once no more chores fit into the current bin, the bin is closed and allocated to one of the remaining agents for whom the last chore fitted. This process continues until all chores are allocated or until no remaining agents can accommodate the chores. If all chores are successfully allocated, the algorithm is said to have *succeeded*; if there are remaining chores that cannot be allocated to any agent, the algorithm *failed*.

We next describe the concept of *First-Fit-Valid*, as defined in (Huang and Segal-Halevi 2023), which will be useful throughout the paper. We note that all the missing proofs can be found in the full version of the paper (Garg, Huang, and Segal-Halevi 2024).

First Fit Valid

We first introduce notations to compare two bundles. For a given a bundle B, let $B[p]$ denote the p-th largest chore in B. If multiple chores share the same cost, we select $B[p]$ according to the universal ordering. We extend this notation to define $v(B[p]) = 0$ when $p > |B|$.

Definition 1 (Lexicographic order)**.** Given a cost function v and two bundles B_1, B_2, we say $B_1 \overset{lex}{=} B_2$ if $v(B_1[p]) = v(B_2[p])$ for all p; $B_1 \overset{lex}{\leq} B_2$ if $B_1 \overset{lex}{=} B_2$ or $v(B_1[q]) < v(B_2[q])$, where q is the smallest index for which $v(B_1[q]) \neq v(B_2[q])$.

Note that every two bundles are comparable by this order.

We define $B_1 \overset{lex}{\geq} B_2$ if $B_2 \overset{lex}{\leq} B_1$; and $B_1 \overset{lex}{<} B_2$ if $B_1 \overset{lex}{\leq} B_2$ and $B_1 \overset{lex}{\neq} B_2$; and similarly define $\overset{lex}{>}$.

Note that, when we consider only a single cost function, we can treat $\overset{lex}{=}$ as $=$. This is because, from the point of view of a single agent, chores with the same cost are equivalent, and can be exchanged anywhere, without affecting any of the arguments in the proofs.

Based on lexicographic order, we can define a *lexicographically maximal* subset.

Definition 2 (Lexicographically maximal subset)**.** Given a set of chores S, a cost function v and a threshold τ, let $S_{\leq\tau} = \{B \subseteq S \mid v(B) \leq \tau\}$ be the set of all bundles with cost at most τ. A bundle B is a *lexicographically maximal subset* of S, w.r.t. v and τ, if $B \in S_{\leq\tau}$ and $B \overset{lex}{\geq} B'$ for any $B' \in S_{\leq\tau}$.

Based on lexicographical maximality, the following notion relates the outputs of the FFD and HFFD algorithms.

Definition 3 (Benchmark bundle)**.** Suppose we are given a set of chores $\mathcal{M}$, a collection of bundles $\mathbf{A}$, a cost function v, and a threshold τ. For each $k \in [n]$, we define the *k-th benchmark bundle* of $\mathbf{A}$, denoted B_k, as a lexicographically maximal subset of $\mathcal{M} \setminus \bigcup_{j<k} A_j$ with respect to v and τ.

Intuitively, once the bundles $A_1, \ldots, A_{k-1}$ have been allocated, the benchmark bundle is the lexicographically maximal feasible subset of the remaining items. The following proposition shows that this is exactly the outcome of running the FFD algorithm for generating the next bundle A_k.

Proposition 1 (Huang and Segal-Halevi 2023)**.** *Let $\mathbf{D}$ be the bundle collection which is output by FFD$(\mathcal{M}, v, \tau)$. Then, for all $i \in [n]$, we have $D_i \overset{lex}{=} B_i$, where B_i is the i-th benchmark bundle of $\mathbf{D}$.*

Next, we show that when using HFFD to generate the bundle in some iteration k, the resulting bundle is guaranteed to be lexicographically greater than or equal to the k-th benchmark bundle. This property is called *First Fit Valid*, which is formally defined below.

Definition 4 (First Fit Valid (FFV))**.** Given a bundle collection $\mathbf{A}$, a cost function v and a threshold τ, we call the tuple $(\mathcal{M}, \mathbf{A}, v, \tau)$ *First Fit Valid* if for any $k \in [n]$, we have $A_k \overset{lex}{\geq} B_k$, where B_k is the k-th benchmark bundle of $\mathbf{A}$, [1] and the lexicographic comparison uses the cost function v.

We emphasize that the definition of a tuple as FFV depends on the entire set of chores $\mathcal{M}$, not only on the chores allocated in $\mathbf{A}$. This is because, if $\mathcal{M}$ contains chores not allocated in $\mathbf{A}$, this may affect the benchmark bundles.

With the notion of First Fit Valid, the next proposition characterizes the output of the HFFD algorithm, where v_ω, τ_ω are the cost function and threshold of the last agent ω in HFFD.

Proposition 2 (Huang and Segal-Halevi 2023)**.** *For any IDO instance $\mathcal{I}$ and any threshold vector $(\tau_1, \ldots, \tau_n)$, if Algorithm HFFD produces a bundle collection $\mathbf{A}$, then for any $\tau \leq \tau_\omega$, the tuple $(\mathcal{M}, \mathbf{A}, v_\omega, \tau)$ is a First Fit Valid tuple.*

1-out-of-$\lfloor \frac{9}{11} n \rfloor$ MMS

In this section, we demonstrate that the HFFD algorithm yields a 1-out-of-$\lfloor \frac{9}{11} n \rfloor$ MMS allocation. Our approach combines the results of FFD for bin packing (Dósa et al. 2013) with a reduction from HFFD to FFD, closely following the method described in (Huang and Segal-Halevi 2023).

Lemma 1. *Let $n \geq 2$ be an integer, $\mathcal{M}$ a set of chores, and v any cost function, Let $d := \lfloor \frac{9}{11} n \rfloor$, and suppose*

$$MMS^d(\mathcal{M}) = 1.$$

If the tuple $(\mathcal{M}, \mathbf{A}, v, 1)$ is First Fit Valid, then the allocation $\mathbf{A}$ must contain all chores in $\mathcal{M}$.

[1] Note that B_k is lexicographically-maximal among all subsets of $\mathcal{M} \setminus \bigcup_{j<k} A_j$ with cost at most τ, but A_k may be lexicographically larger than B_k since its cost may be larger than τ.

Proof. We assume by contradiction that there is a chore $c^- \in \mathcal{M}$, that is not in any bundle in $\mathbf{A}$.

Let us consider the case where $v(c^-) \leq \frac{2}{11}$. Given that $\text{MMS}^d(\mathcal{M}) = 1$, the total cost of all chores in $\mathcal{M}$ is at most nd. This implies there exists at least one bundle A_j whose cost is less than $d \leq \frac{9}{11}n$. Consequently, we have $v(A_j \cup \{c^-\}) \leq 1$. However, $A_j \cup \{c^-\}$ is lexicographically larger than A_j, thus contradicting the definition of First Fit Valid.

We can now assume that $v(c^-) > \frac{2}{11}$. Because removing all chores less than c^- would not affect the First Fit Valid. This operation is dicussed in Tidy-up Procedure in paper (Huang and Segal-Halevi 2023).

We will construct another instance $\mathcal{M}'$ by decreasing the cost of some chores and removing some chores from $\mathcal{M}$. Then we can obtain another allocation $\mathbf{A}'$, which is output of FFD$(\mathcal{M}', v, 1)$. The unallocated chore c^- will still be in $\mathcal{M}'$, and not in the output of FFD. Notice that for this new instance the maximin share for $\lfloor \frac{9}{11}n \rfloor$ bins is no more than 1. Then we have a contradiction here. Because FFD algorithm can allocate all chores in this case (Dósa et al. 2013).

We now show how to construct such an instance, using notions from (Huang and Segal-Halevi 2023).

We first define *redundant chores*. Given a bundle A_j, a chore $A_j[p]$ is called *redundant* if $p > k$ where k is the smallest integer such that $v(\cup_{p \leq k} A_j[p]) \geq 1$. We remove all redundant chores from $\mathcal{M}$ and $\mathbf{A}$ at the beginning. We keep the notation $\mathcal{M}$ and $\mathbf{A}$, but assume that there is no redundant chores in the rest of the proof.

To see why they are called redundant, we can go back to the algorithm. If we set threshold 1 for FFD, then FFD will not allocate to any bundle chores with total cost more than 1. So removing redundant chores does not influence the fact that there is an unallocated chore.

Let k be the smallest integer such that $v(A_k) > 1$. Apply Proposition 1 to the first $k-1$ bundles of $\mathbf{A}$, they are the same as the output of FFD$(\mathcal{M}, v, 1)$. To fix the k-th bundle, we introduce the following concept from (Huang and Segal-Halevi 2023).

Definition 5 (Suitable reduced chore)**.** Let $(\mathcal{M}, \mathbf{A}, v, 1)$ be a First Fit Valid tuple without any redundant chores. Let k be the smallest integer such that $v(A_k) > 1$. Let c_k^* be the smallest chore in A_k. Let c_k° be a chore with $v(c_k^\circ) < v(c_k^*)$ and $\mathcal{M}' = \mathcal{M} \setminus \{c_k^*\} \cup \{c_k^\circ\}$.
Let $\mathcal{P}$ be the output of $FFD(\mathcal{M}', v, 1)$. The chore c_k° is a *suitable reduced chore* of c_k^* if

$$\bigcup_{j \leq k} P_j = \left(\bigcup_{j \leq k} A_j \right) \setminus \{c_k^*\} \cup \{c_k^\circ\}. \tag{1}$$

In other words: replacing c_k^* by c_k° has no effect on the allocation of bundles $i > k$.

If we can always find a suitable reduced chore, then we can fix the allocation to the output of FFD algorithm one bundle by one bundle. The remaining thing is to prove there is a suitable reduced chore.

Roughly, when we want to find a suitable reduced chore for bundle A_k, its reduced cost should be such that the new cost of A_k is at most 1. This puts an upper bound on the cost of suitable reduced chores. This motivates the following definition.

Definition 6 (Fit-in space (Huang and Segal-Halevi 2023))**.** Given an index k, let c_k^* be the last (smallest) chore in bundle A_k. The *fit-in space* for A_k is $ft(k) := 1 - v(A_k \setminus \{c_k^*\})$.

Note that since there are no redundant chores, the fit-in space is non-negative. The following claim is useful for simplifying our proof.

Claim 1 (Huang and Segal-Halevi 2023 (Lemmas 11 and 12))**.** *Let $\gamma = \frac{2}{11}$. If there are no redundant chores, and no chore is smaller than the unallocated chore c^-, and $ft(k) \leq 2\gamma = \frac{4}{11}$, then a suitable reduced chore exists.*

Now we only need to consider the case of a large fit-in space $ft(k) > 2\gamma = \frac{4}{11}$. This is the main point in which our proof differs from theirs. In their proof, they used a procedure called "Tidy-Up", which makes the instance easier to work with. We cannot use this procedure here, as it might reduce the number of bundles. Therefore, we need a whole different proof for the case of a large fit-in space. Let c_k^* be the smallest chore in A_k.

We prove that, for the case of large fit-in space, there are exactly 2 chores in bundle A_k.

No chore could be larger than 1, as this is the maximin share for $\lfloor \frac{9}{11}n \rfloor$ bins. On the other hand, there cannot be three chores in bundle A_k. Otherwise, it means that there are at least two chores except c_k^*. When the cost $v(A_k) > 1$, the smallest chore is larger than the fit-in space by definition. So the cost of each chore in A_k is at least $\frac{4}{11}$. So the total cost of $A_k \setminus \{c_k^*\}$ is at least $2 \cdot \frac{4}{11} = \frac{8}{11}$. By the definition of fit-in space, we have $v(A_k \setminus \{c_k^*\}) = 1 - ft(k) < \frac{7}{11}$. We have a contradiction here. Therefore, bundle A_k contains exactly 2 chores, namely $A_k[1]$ and $A_k[2] = c_k^*$.

Let c_k° be a chore with cost $v(c_k^\circ) = ft(k)$. We prove it is a suitable reduced chore. Let us run FFD for the first k bundles on the instance after replacing chore c_k^* with c_k°.

Suppose first that c_k° is not allocated to a bundle P_i with $i < k$. By selection of k, this means that the first $k-1$ bundles generated by FFD are equal to those of $\mathbf{A}$ ($P_i = A_i$ for all $i < K$), then when filling P_k, the chores $A_k[1]$ and c_k° are available. Because $\mathbf{A}$ is FFV, $A_k[1]$ must be the largest remaining chore, so it will be allocated to P_k first. Now, the remaining room in P_k is exactly equal to the fit-in space, which is the size of c_k°. Therefore, c_k° will be allocated to P_k, and we are done.

Now we consider the case that c_k° is allocated to a bundle P_i where $i < k$. We have the following claim.

Claim 2. *Suppose that c_k° is allocated to a bundle P_i where $i < k$, we have $v(P_i[1]) = v(P_k[1]) = v(A_k[1]) = v(P_j[1])$ for all $i < j < k$.*

Proof. We first show $v(A_i[1]) = v(P_i[1])$ when $i \leq k$. As allocation $\mathbf{A}$ is First Fit Valid w.r.t. $\mathcal{M}$, and the costs of first

$k-1$ bundles are no more than 1, the first $k-1$ bundles in $\mathbf{A}$ can be considered as the output of FFD algorithm on $\mathcal{M}$.

As FFD processes large chores before small chores, any chore larger than c_k^* that is allocated in A_j where $j < k$, will be allocated in P_j, as its allocation is not influenced by c_k^*.

We also have $v(A_k[1]) = v(P_k[1])$ by FFV, as it is the largest chore remaining after removing the chores in the first $k-1$ bundles.

By the properties of FFD, we have $v(P_i[1]) \geq v(P_k[1]) = v(A_k[1]) = 1 - ft(k)$, as $A_k[1]$ is the only chore in A_k except the last one.

If c_k° is allocated to P_i for $i < k$, we have $v(P_i[1] \cup c_k^\circ) \leq 1$, which means $v(P_i[1]) \leq 1 - v(c_k^\circ) = 1 - ft(k)$.

Combining the two inequalities on $v(P_i[1])$ leads to $v(P_i[1]) = 1 - ft(k) = v(A_k[1])$. As $v(P_k[1]) \leq v(P_j[1]) \leq v(P_i[1])$ for $i < j < k$, the costs of all these chores must be equal. ■

With Claim 2, we have an easy description of what happened between $\mathcal{P}$ and $\mathcal{A}$. We prove that there will be a shift: for all $i < j \leq k$, we have $P_j = \{A_j[1]\} \cup (A_{j-1} \setminus \{A_{j-1}[1]\})$. First, we have $P_i = \{A_i[1]\} \cup \{c_k^\circ\}$. Basically, the argument is as follows: If you give the same space and same set of chores, the FFD algorithm will get the same result. We prove that it is true for $i+1$. The similar argument directly works for other indexes. Let S be the set of chores $A_i \setminus \{A_i[1]\}$. The set of chores S are allocated into a space $1 - v(A_i[1]) = ft(k)$. After allocating the chore $P_{i+1}[1]$, the remaining space is $ft(k)$. When FFD algorithm try to generate P_{i+1}, let us consider the set of chores which are no larger than $ft(k)$. Let us denote the set by $L = \{c \mid \mathcal{M}' \setminus (\cup_{j \leq i} P_i)\}$. By Proposition 1, the first i bundles of allocation $\mathbf{A}$ are the same to the output of FFD$(\mathcal{M}, v, 1)$. Let us consider the set of chores which are no larger than $ft(k)$ when A_i is generated. Let us denote the set by $R = \{c \mid v(c) \leq ft(k) \text{ and } c \in \mathcal{M} \setminus (\cup_{j<i} A_j)\}$. We have $R = L$. So what are allocated to A_i will be also allocated to P_{i+1}. For other bundles, the argument is similar.

The chores in the bundle with index later than k will not be affected the allocation of first k bundle. So c_k° is a suitable reduced chore. □

Theorem 1. *Suppose that the MMS value for $\lfloor \frac{9}{11} n \rfloor$ bundles (bins) is 1 for every agent. If the HFFD algorithm is executed with a threshold of at least* 1 *for each agent, then all chores are allocated.*

Proof. Let $\mathbf{A}$ be the output of HFFD. By Proposition 2, since $\tau_\omega \geq 1$, the tuple $(\mathcal{M}, \mathbf{A}, v_\omega, 1)$ is First-Fit-Valid.

Given that the 1-out-of-$\lfloor \frac{9}{11} n \rfloor$ MMS of v_ω is 1 by assumption, we can apply Lemma 1 with $v = v_\omega$, and thus conclude that $\mathbf{A}$ must contain all chores. □

Reduction: Factored & Bivalued Instances

We will demonstrate that for factored and personalized-bivalued instances, the approximation ratio of HFFD is equal to that of MultiFit.

To facilitate our proof, we will use a swapping operation extensively. The swapping process is defined as follows:

Definition 7. Given an allocation $\mathbf{A}$, a set of items $T_i \subseteq A_i$ and $T_j \subseteq A_j$, let allocation $\mathbf{A}'$ be the allocation after swapping T_i and T_j. Formally, we have $A'_k = A_k$ if $k \neq \{i, j\}$ and $A'_i = (A_i \setminus T_i) \cup T_j$ and $A'_j = (A_i \setminus T_j) \cup T_i$. We define

$$\textit{SWAP}\,(\mathbf{A}, T_i, T_j) = \mathbf{A}' \ .$$

When the subscripts i and j are not crucial or their meaning is clear from the context, we will use symbols without subscripts. Additionally, there may be cases where one set of chores could be empty. For example, the operation $\textit{SWAP}\,(\mathcal{A}, \emptyset_j, T)$ denotes moving the set of chores T to bundle A_j. The subscript j specifies the bundle receiving the chores in T.

Lemma 2 (Reduction Lemma)**.** *For any threshold $\tau > 0$, and any cost function v that is either factored or bivalued if FFD$(\mathcal{M}, v, \tau)$ allocates all the chores into n bins, and $(\mathcal{M}, \mathcal{Q}, v, \tau)$ is a First-Fit Valid tuple, then $\mathcal{Q}$ contains all chores in $\mathcal{M}$.*

We will prove the above lemma for factored cost functions in Section and for bivalued cost functions in Section 17.

This lemma implies two things:

Corollary 1. *For both factored and personalized-bivalued instances, we can reduce multiple agents case to single agent case. That is, the approximation ratio of HFFD is equal to that of MultiFit.*

Proof. This follows from Lemma 2 as the outcome of HFFD when all agents have threshold τ is FFV for v_ω and τ. □

Corollary 2. *For both factored and personalized-bivalued instances, FFD is* monotone *in the following sense: if FFD$(\mathcal{M}, v, \tau)$ allocates all chores, then FFD$(\mathcal{M}, v, \beta)$ allocates all chores for any $\beta > \tau$.*

Proof. This follows from Lemma 2 as the outcome of FFD$(\mathcal{M}, v, \beta)$ is First Fit Valid for v and τ. □

Polynomial-time computation Corollary 2 implies that we can use binary search to design polynomial-time algorithms for factored and personalized bivalued instances. The process involves performing a binary search to determine a minimal threshold τ_i for each agent i, such that FFD$(\mathcal{M}, v_i, \tau_i)$ can allocate all chores. These thresholds τ_i are then used in the HFFD algorithm. By Corollary 2, the HFFD algorithm will allocate all the chores, ensuring that each agent i gets a bundle no larger than τ_i.

However, this approach does not extend to the general case, as discussed in (Huang and Segal-Halevi 2023). In general, combining all thresholds into the HFFD algorithm may not guarantee the allocation of all chores.

Factored Instance

A cost function is called *factored* if any smaller cost is a factor of the next-larger cost, i.e., if $v(c_m) \leq \cdots \leq v(c_1)$ then $v(c_m) \mid v(c_{m-1}) \mid \cdots \mid v(c_2) \mid v(c_1)$. The notion first appeared with relation to bin packing in (Coffman, Garey, and Johnson 1987), where it is called "divisible item sizes". It was introduced into fair division in (Ebadian, Peters, and Shah 2021).

Claim 3 ((Coffman, Garey, and Johnson 1987)). *For a factored instance, given an index j and a set S of indices such that*
(1) $v(c_j) \geq v(c_i)$ for any $i \in S$ and
(2) $v(S) \geq v(c_j)$,
there is a subset $S' \subseteq S$ such that $v(S') = v(c_j)$.

Theorem 2 ((Coffman, Garey, and Johnson 1987)). *In job scheduling with a factored cost function, FFD with threshold exactly equal to the minimum makespan always allocates all jobs.*

We will now extend Theorem 2 from FFD to HFFD. We will use the following reduction lemma.

Proof of Reduction Lemma for factored instances. Here is a high-level idea for the proof. Given an allocation $\mathcal{P}$ produced by the FFD algorithm and a First-Fit Valid allocation $\mathcal{Q}$, our goal is to transform $\mathcal{P}$ into $\mathcal{Q}$ using a series of swapping operations.

We describe the swapping operations in Algorithm 1 and provide an illustrative example below.

Example 1. Suppose the threshold is 10 and $P_1 = \{a, b, c, d, e\}$ with chore costs $4, 2, 2, 1, 1$, and $Q_1 = \{w, x, y, z\}$ with chore costs $4, 4, 4, 1$. Note that the instance is factored and $Q_1 \overset{lex}{>} P_1$. At the inner for loop, the condition $v(Q_k[j]) > v(P_k[j])$ is first satisfied for $j = 2$. The set $T = \{b, c, d, e\}$ and its total cost is 6 which is larger than $v(Q_k[2])$. We find the set $S = \{b, c\}$, remove it from P_1 and swap it with $\{x\}$. Now we have $P_1 = \{a, x, d, e\}$ with chore costs $4, 4, 1, 1$.

Next, after the swap, $v(Q_k[j]) > v(P_k[j])$ for $j = 3$. We have $T = \{d, e\}$, its total cost is 2 which is now smaller than $v(Q_k[j])$. So we remove T from P_1 and swap it with $\{y\}$. The new $P_1 = \{a, x, y\}$ with chore costs $4, 4, 4$.

Next, for $j = 4$ we have $v(Q_k[j]) > v(P_k[j])$ as $v(P_k[j]) = 0$ by definition. $T = \emptyset$, and we swap it with z. So the final P_1 is $\{a, x, y, z\}$, which is lex-equivalent to Q_1.

Claim 4. *At each iteration k of the main loop, the cost of any bundle P_i with $i > k$ does not increase.*

Proof. The proof is by induction. The base $k = 0$ is straightforward. For $k \geq 1$, assume that the claim holds for all iterations before k. We have $v(P_k) \leq \tau$ since τ is the threshold used by FFD. By the First-Fit Valid property, if we check the chores of P_k and Q_k from largest to smallest, the first different place j must satisfy $v(Q_k[j]) > v(P_k[j])$.

In the swap of line 10, the swap is between two sets of equal cost, $v(S) = v(Q_k[j])$, so the costs of all bundles to not change.

In the swap of line 13, the swap is between sets with different costs, $v(T) < v(Q_k[j])$, so the cost of P_k increases and the cost of some bundle P_i with $i > k$ decreases. □

By Claim 4, at the start of each iteration k, $v(P_k) \leq \tau$, so by First Fit Valid definition, it still holds that $Q_k \overset{lex}{\geq} P_k$.

Algorithm 1: Reduction by swapping for a factored cost function

Data: $\mathcal{Q}$ a First-Fit Valid allocation; $\mathcal{P}$ output of FFD on same items.

```
1  for k=1 to N do
2      if Q_k =^lex P_k then
3          Continue to next k.
4      end
       /* Else, Q_k >^lex P_k by FFV          */
       /* Compare Q_k with P_k from
          largest to smallest chore           */
5      for j= 1 to |Q_k| do
6          if v(Q_k[j]) > v(P_k[j]) then
7              Let T = ∪_{w≥j} P_k[w];
8              if v(T) ≥ v(Q_k[j]) then
9                  Find a set S ⊆ T such that
                     v(S) = v(Q_k[j]);
                   /* Claim 3 guarantees
                      existence of S          */
10                 Update P = SWAP(P, S, Q_k[j])
11             end
12             else
13                 Update P = SWAP(P, T, Q_k[j])
14             end
15         end
16     end
17 end
```

During iteration k, after each iteration j of the inner loop, the set $P_k[1, \ldots, j]$ is leximin-equivalent to $Q_k[1, \ldots, j]$. If, for some $j \leq |Q_k|$, we have $|P_k| < j$, then $T = \emptyset$, and the algorithm will move some chores from some later bundles into P_k. Thus, at the end of iteration k, we will have $P_k \overset{lex}{=} Q_k$. Consequently, the entire process ensures that $\mathcal{P} \overset{lex}{=} \mathcal{Q}$. As allocation $\mathcal{P}$ contains all chores, allocation $\mathcal{Q}$ must also include all chores. This completes the proof of Lemma 2 for factored instances. □

Theorem 3. *In chore allocation with factored cost functions, HFFD with the thresholds of all agents equal to their maximin share always allocates all chores.*

Proof. This follows from Theorem 2 and Corollary 1. □

Bivalued Instance

In this section, we consider the case where v is a bivalued cost-function. The reduction process for this case is described in Algorithm 2, and we provide an illustrative example below. For any bundle A, we denote by $L(A)$ the set of large chores in A, and by $S(A)$ the set of small chores in A.

Example 2. Suppose the threshold is 20 and $P_1 = \{a, b, c, d, e\}$ with chore costs $7, 7, 2, 2, 2$ and $Q_1 = \{u, v, w, x, y, z\}$ with chore costs $7, 7, 7, 7, 2, 2$. We have $|L(Q_1)| = 4 > 2 = L(P_1)$, so there must be a large chore in some bundle P_i with $i > 1$; we take the last such chore

Algorithm 2: Reduction by swapping for a bivalued cost function

Data: $\mathcal{Q}$ a First-Fit Valid allocation; $\mathcal{P}$ output of FFD on same items.

for k=*1* **to** N **do**
  if $Q_k \overset{lex}{=} P_k$ **then**
    Continue to next k.
  end
  /* Else $Q_k \overset{lex}{>} P_k$ by FFV, so either $|L(Q_k)| > |L(P_k)|$, or $|L(Q_k)| = |L(P_k)|$ and $|S(Q_k)| > |S(P_k)|$. */
  if $|L(Q_k)| > |L(P_k)|$ **then**
    Let cl be the large chore which appears last in allocation $\mathcal{P}$;
    /* We prove later that chore cl must be in P_i for $i > k$ */
    Update $\mathcal{P} = \mathit{SWAP}(\mathcal{P}, cl, S(P_k))$;
  end
  /* At this point we can assume wlog that $Q_k \supseteq P_k$. */
  for $c \in Q_k \setminus P_k$ **do**
    Let chore cl be the one that appears last in allocation $\mathcal{P}$ with the same cost as c;
    Update $\mathcal{P} = \mathit{SWAP}(\mathcal{P}, cl, \emptyset_k)$;
  end
end

(say x) and swap it with c, d, e, so now $P_1 = \{a, b, x\}$ with chore costs $7, 7, 7$.

Now, we can assume w.l.o.g. that the large chores in P_1 are the same as some large chores in Q_1, say u, v, x, so $P_1 = \{u, v, x\}$ and $P_1 \subseteq Q_1$. In the next loop, we simply add to P_1 the missing chores w, y, z or some other chores with the same costs.

Claim 5. *1. In iteration k, the cost of any bundle P_i with $i > k$ does not increase.*

2. If initially the number of large chores in Q_k is greater than in P_k, then the number of large chores in P_k does not change in iterations 1 to $k-1$.

As in the factored case, at the end of iteration k, we have $P_k \overset{lex}{=} Q_k$, so at the end of the algorithm $\mathcal{P} \overset{lex}{=} \mathcal{Q}$. As $\mathcal{P}$ contains all chores, $\mathcal{Q}$ contains all chores too. This completes the proof of Lemma 2 for bivalued instances.

Following Corollary 1, in order to prove an approximation ratio of HFFD on bivalued instances, it suffices to prove an approximation ratio of MultiFit. We are not aware of previous work analyzing the performance of MultiFit on bivalued instances. In this section, we establish a tight approximation ratio of $15/13$.

Throughout this section, we denote the large chore size by l, the small chore size by s, and the MMS value by μ.

Lower bound

The lower bound of $15/13$ is shown using a simple example.

Example 3. There are $n = 3$ agents. There are three large chores with cost $l = 4$, and nine small chores with cost $s = 3$. The MMS is 13, as the chores can be partitioned into three bundles containing 1 large chore and 3 small chores each. However, FFD with any bin size in $[13, 15)$ packs the three large chores into a single bin, and eight small chores into the following two bins, so one small chore remains unallocated.

Upper bound

Our proof for the upper bound uses a technique similar to that of Section . We establish the following theorem using an involved case analysis, given in the full version (Garg, Huang, and Segal-Halevi 2024).

Theorem 4. *For personalized bivalued instances of chores, the HFFD algorithm achieves a tight approximation ratio of $\frac{15}{13}$ for MMS.*

Proof sketch. Algorithm 3 (in the full version of the paper (Garg, Huang, and Segal-Halevi 2024)) accepts as input an allocation $\mathcal{Q}$ with maximum cost μ, and an allocation $\mathcal{P}$ generated by FFD with threshold $\tau := \frac{15}{13}\mu$. The algorithm processes both allocations bin by bin. In each iteration k, it modifies $\mathcal{Q}$ such that Q_k becomes equal to P_k. The modification is done by swapping some chores in Q_k with some chores from bundles Q_z for $z > k$. Using a detailed case analysis we show that, even after the swap, the cost of Q_z remains at most τ. At the end of iteration n, $Q_i = P_i$ for all $i \in [n]$. As $\mathcal{Q}$ contains all chores, this implies that $\mathcal{P}$ allocates all chores in n bins, which concludes the proof. □

Conclusion

Huang and Lu (2021) first explored the connection between the bin packing problem and the fair allocation of chores, an approach that was further utilized in (Huang and Segal-Halevi 2023). Building on this foundation, our paper delves deeper into this connection. We demonstrated that the HFFD algorithm could improve the state-of-the-art of maximin share (MMS) in factored instances, personalized bivalued instances, and 1-out-of-d MMS allocations. The central question we explore is: Can we discover additional connections between bin packing and the fair allocation of chores? This question could possibly extend beyond the scope of chore allocation and has the potential to inspire algorithmic solutions for the allocation of goods as well.

An intriguing open question in this area is whether the HFFD (and FFD) algorithm is monotonic in a certain range. In this paper, we demonstrate the monotonicity of the algorithm for two special classes of chores costs. This question is closely tied to the possibility of designing an efficient algorithm for MMS allocations that achieves the exact approximation ratio of the bin packing problem. The monotonicity in the general case remains unclear. Another open problem is whether MMS allocations always exist for personalized bivalued instances. To the best of our knowledge, there are no known counterexamples demonstrating the non-existence of MMS for these instances.

Acknowledgments

Jugal Garg was supported by NSF Grants CCF-1942321 and CCF-2334461. Xin Huang was supported by JST ERATO Grant Number JPMJER2301, Japan. Erel Segal-Halevi was funded by Israel Science Foundation grant no. 712/20.

References

Aigner-Horev, E.; and Segal-Halevi, E. 2022. Envy-free matchings in bipartite graphs and their applications to fair division. *Information Sciences*, 587: 164–187.

Akrami, H.; and Garg, J. 2024. Breaking the 3/4 Barrier for Approximate Maximin Share. In *Proc. 35th Symp. Discrete Algorithms (SODA)*, 74–91.

Akrami, H.; Garg, J.; Sharma, E.; and Taki, S. 2023. Simplification and Improvement of MMS Approximation. In *Proc. 32nd Intl. Joint Conf. Artif. Intell. (IJCAI)*.

Akrami, H.; Garg, J.; Sharma, E.; and Taki, S. 2024. Improving Approximation Guarantees for Maximin Share. In *Proc. 25th Conf. Economics and Computation (EC)*.

Amanatidis, G.; Aziz, H.; Birmpas, G.; Filos-Ratsikas, A.; Li, B.; Moulin, H.; Voudouris, A. A.; and Wu, X. 2023. Fair Division of Indivisible Goods: Recent Progress and Open Questions. *Artificial Intelligence*, 322: 103965.

Amanatidis, G.; Markakis, E.; Nikzad, A.; and Saberi, A. 2017. Approximation Algorithms for Computing Maximin Share Allocations. *ACM Transactions on Algorithms (TALG)*, 13(4): 1–28.

Aziz, H.; Biró, P.; Lang, J.; Lesca, J.; and Monnot, J. 2019. Efficient reallocation under additive and responsive preferences. *Theor. Comput. Sci.*, 790: 1–15.

Aziz, H.; Rauchecker, G.; Schryen, G.; and Walsh, T. 2017. Algorithms for max-min share fair allocation of indivisible chores. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, 335–341.

Barman, S.; and Krishnamurthy, S. K. 2020. Approximation Algorithms for Maximin Fair Division. *ACM Transactions on Economics and Computation (TEAC)*, 8(1): 1–28.

Budish, E. 2011. The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes. *Journal of Political Economy*, 119(6): 1061 – 1103.

Coffman, E. G., Jr.; Garey, M. R.; and Johnson, D. S. 1978. An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, 7(1): 1–17.

Coffman, E. G., Jr.; Garey, M. R.; and Johnson, D. S. 1987. Bin packing with divisible item sizes. *Journal of Complexity*, 3(4): 406–428.

Dósa, G.; Li, R.; Han, X.; and Tuza, Z. 2013. Tight absolute bound for First Fit Decreasing bin-packing: FFD(L) $\leq$ 11/9 OPT(L) + 6/9. *Theor. Comput. Sci.*, 510: 13–61.

Ebadian, S.; Peters, D.; and Shah, N. 2021. How to fairly allocate easy and difficult chores. *arXiv preprint arXiv:2110.11285*.

Feige, U. 2022. Maximin fair allocations with two item values. https://www.wisdom.weizmann.ac.il/feige/mypapers/MMSab.pdf. Accessed: 2022-02-22.

Feige, U.; Sapir, A.; and Tauber, L. 2021. A Tight Negative Example for MMS Fair Allocations. In *Proc. 17th Conf. Web and Internet Economics (WINE)*, 355–372.

Garg, J.; Huang, X.; and Segal-Halevi, E. 2024. Improved Maximin Share Approximations for Chores by Bin Packing. *arXiv:2411.04391*.

Garg, J.; and Taki, S. 2021. An Improved Approximation Algorithm for Maximin Shares. *Artificial Intelligence*, 103547.

Gates, V.; Griffiths, T. L.; and Dragan, A. D. 2020. How to Be Helpful to Multiple People at Once. *Cognitive Science*, 44.

Ghodsi, M.; HajiAghayi, M.; Seddighin, M.; Seddighin, S.; and Yami, H. 2018. Fair Allocation of Indivisible Goods: Improvements and Generalizations. In *Proc. 19th Conf. Economics and Computation (EC)*, 539–556.

Hochbaum, D. S.; and Shmoys, D. B. 1987. Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, 34(1): 144–162.

Hosseini, H.; and Searns, A. 2021. Guaranteeing Maximin Shares: Some Agents Left Behind. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, 238–244.

Hosseini, H.; Searns, A.; and Segal-Halevi, E. 2021. Ordinal Maximin Share Approximation for Goods. *J. Artif. Intell. Res.*, 74.

Hosseini, H.; Searns, A.; and Segal-Halevi, E. 2022. Ordinal Maximin Share Approximation for Chores. In *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022*, 597–605.

Hosseini, H.; Sikdar, S.; Vaish, R.; and Xia, L. 2021. Fair and Efficient Allocations under Lexicographic Preferences. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*, 5472–5480. AAAI Press.

Huang, X.; and Lu, P. 2021. An Algorithmic Framework for Approximating Maximin Share Allocation of Chores. In *EC '21: The 22nd ACM Conference on Economics and Computation*, 630–631.

Huang, X.; and Segal-Halevi, E. 2023. A reduction from chores allocation to job scheduling. In *Proc. 24th Conf. Economics and Computation (EC)*.

Johnson, D. S. 1973. *Near-optimal bin packing algorithms*. Phd thesis, Massachusetts Institute of Technology.

Kurokawa, D.; Procaccia, A. D.; and Wang, J. 2018. Fair Enough: Guaranteeing Approximate Maximin Shares. *J. ACM*, 65(2): 8:1–8:27.

Liu, S.; Lu, X.; Suzuki, M.; and Walsh, T. 2024. Mixed Fair Division: A Survey. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 22641–22649.