Multi-agent Online Scheduling: MMS Allocations for Indivisible Items

Shengwei Zhou [1] Rufan Bai [1] Xiaowei Wu [1]

Abstract

We consider the problem of fairly allocating a sequence of indivisible items that arrive online in an arbitrary order to a group of n agents with additive normalized valuation functions, we consider both the allocation of goods and chores and propose algorithms for approximating maximin share (MMS) allocations. When agents have identical valuation functions the problem coincides with the semi-online machine covering problem (when items are goods) and load balancing problem (when items are chores), for both of which optimal competitive ratios have been achieved. In this paper, we consider the case when agents have general valuation functions. For the allocation of goods, we show that no competitive algorithm exists even when there are only three agents and propose an optimal 0.5-competitive algorithm for the case of two agents. For the allocation of chores, we propose a $(2-1/n)$-competitive algorithm for $n \geq 3$ agents and a $\sqrt{2} \approx 1.414$-competitive algorithm for two agents. Besides, we show that no algorithm can do better than $15/11 \approx 1.364$-competitive for two agents.

1. Introduction

Traditional machine learning algorithms usually focus on global objectives such as efficiency or maximizing profit and have no guarantee of fairness between individuals. As learning decision-making is increasingly involved in our daily life, the problem of algorithm bias has received increasing attention. Motivated by several real-world problems in which decision processes impact human beings who must be treated fairly and unbiasedly, there is increasing attention on fair learning algorithms (Chen et al., 2019; Backurs et al., 2019; Li et al., 2021). We focus on the online scheduling problem where jobs arrive online and each incoming job should be assigned by the algorithm immediately. Contrast to the classic online scheduling models that focus on allocation to machines and object on optimizing global objectives, e.g., maximizing the minimum load (Tan & Wu, 2007; Max et al., 2022; Cheng, 2022), or minimizing the maximum load (Kellerer et al., 2015), we consider the allocation to agents and study the online scheduling problem in a multi-agent perspective. The problem coincides with the online maximin share (MMS) allocation problem proposed by Amanatidis et al. (2022), which falls into the class of fair allocation problems.

In the fair allocation problem, there is a set M of m indivisible items (jobs) and a group N of n (heterogeneous) agents, where each agent $i \in N$ has a valuation function v_i on the items. For indivisible items, each item $e \in M$ must be allocated to exactly one of the agents in N. Therefore each allocation corresponds to a partitioning of the items into n bundles $(X_1, \ldots, X_n)$, where agent $i \in N$ receives bundle X_i. When agents have positive values on the items, we call the items *goods*, e.g., consider the allocation of gifts to kids; when agents have negative values on the items, we call the items *chores*, e.g., when allocating tasks to workers. In this paper, we study both the allocation of goods and chores. For the case of chores, we assume that agents have positive *costs* on the items and refer to the valuation function of agent i as a cost function $c_i : 2^N \to \mathbb{R}^+$.

Different fairness notions have been proposed to measure how fair an allocation is, e.g., the envy-freeness (EF) (Foley, 1967; Lipton et al., 2004; Caragiannis et al., 2019), proportionality (PROP) (Steihaus, 1948) and maximin fair share (MMS) (Budish, 2011). In this paper, we focus on the fairness notion of MMS. Informally speaking, the MMS value MMS_i of an agent $i \in N$ is the best she can guarantee if she gets to partition items into n bundles but is the last agent to pick a bundle. An allocation is called MMS if every agent receives a bundle with objective no worse than her MMS value. For the case of goods, that means $v_i(X_i) \geq \mathsf{MMS}_i$ for all $i \in N$; for the case of chores, that means $c_i(X_i) \leq \mathsf{MMS}_i$ for all $i \in N$. Similar to several works on online scheduling (Ebenlendr et al., 2005; Wu et al., 2007; Angelelli et al., 2007; Cheng et al., 2005; Lee & Lim, 2013), we consider the setting of semi-online in

[1]IOTSC, University of Macau, Macao SAR, China. This project is funded by the Science and Technology Development Fund (FDCT), Macau SAR (file no. 0014/2022/AFJ, 0085/2022/A, 0143/2020/A3 and SKL-IOTSC-2021-2023) . Correspondence to: Xiaowei Wu <xiaoweiwu@um.edu.mo>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

which algorithms are given partial information before items (jobs) arrive. We assume the sum of the total value of items (the total processing time of jobs) is known, which can be modeled as all valuation functions (resp. cost functions) are normalized, i.e. $v_i(M) = n$ (resp. $c_i(M) = n$) for all $i \in N$. Hence for allocating goods, the approximation ratio with respect to MMS is at most 1 while for chores it is at least 1. The partial information can be considered *learned information* based on the historical data. For example, in the food bank problem proposed by Walsh (2014), the food bank has to distribute donated food to charities in an online manner. While the amount of food is unknown on an hourly or daily basis, the total amount in a fixed period (e.g. a week) can often be accurately estimated based on historical data. To name another example, consider a manufacturing company, e.g. Foxconn, that receives online manufacturing orders and needs to assign orders to different working units in a fair way. Again, while the daily order volume may fluctuate greatly, the total volume in a month is often stable and predictable.

The problem of online approximation of the MMS allocations for agents with normalized identical valuations coincides with the *semi-online machine covering* problems (Ebenlendr et al., 2005; Wu et al., 2007) when items are goods, and the *semi-online load balancing* problem (Angelelli et al., 2007; Cheng et al., 2005; Lee & Lim, 2013) when items are chores, in the research field of online scheduling (where the items are jobs and agents are machines). Specifically, for allocating goods, the common MMS of agents corresponds to the minimum load of the machines in the optimal scheduling; for chores, it corresponds to the maximum load of the machines, e.g., the makespan, in the optimal scheduling. For $n = 2$ agents, Kellerer et al. (1997) present a $2/3$-competitive algorithm for goods and a $4/3$-competitive algorithm for chores and show that these competitive ratios are the best possible. For $n \geq 3$ agents, Tan & Wu (2007) propose a $1/(n-1)$-competitive algorithm and show that it is optimal for the allocation of goods; Kellerer et al. (2015) give a 1.585-competitive algorithm for the allocation of chores, which is also optimal due to the lower bound by Albers & Hellwig (2012).

1.1. Our Results

In this paper, we consider both the allocation of goods and chores and present upper and lower bounds for approximating MMS allocations. As in (Gkatzelis et al., 2021; Bogomolnaia et al., 2022; Barman et al., 2022), we assume that the valuation functions are normalized[1], e.g., $v_i(M) = n$ for all $i \in N$ for goods and $c_i(M) = n$ for all $i \in N$ for chores. We refer to an online algorithm as r-*competitive* if, for any online instance, the allocation returned by the algorithm is always r-approximate MMS. We only consider deterministic algorithms in this paper.

Due to the unknown future, the online setting brings many challenges to the fair allocation problem and most of the classic algorithms cease to work. There are two main difficulties in designing online algorithms: (1) the irrevocable decision-making, and (2) the arbitrary arrival order of items. For example, the *envy-cycle elimination* (Lipton et al., 2004) algorithms heavily rely on exchanging items among agents to improve the allocation, which is not allowed in the online setting, as the allocation decisions are irrevocable. Other classic algorithms for approximating MMS allocations (Aziz et al., 2022; 2017; Huang & Lu, 2021) are built on the reduction to identical ordering instances and allocate items in decreasing order of values/costs. Unfortunately in the online setting, the algorithm has no control over the arrival order of items and all these algorithms fail to work.

To get around these difficulties, we combine the ideas from the fair allocation field with that from the online scheduling field and propose (deterministic) competitive online algorithms that work against adaptive adversaries, for both goods and chores. For the allocation of goods, we show that no algorithm has a competitive ratio strictly larger than 0 with respect to MMS, even when there are only three agents. In contrast, we show that competitive algorithms exist for $n = 2$ agents by proposing a 0.5-competitive algorithm, and show that this is optimal for any online algorithms. We also present a 0.5-competitive algorithm for a general number of agents under the assumption that the items arrive in the order from the most valuable to the least valuable. We further consider the small goods instances in which the value of each item (to each agent) is bounded by some $\alpha < 1$ and present a $(1-\alpha)$-competitive algorithm for general number of agents. Due to the page limit, the proofs of the above results are deferred to the full version. Then we turn to the allocation of chores and propose a $(2-1/n)$-competitive algorithm for n agents, which gives a 1.5-competitive algorithm for the case of $n = 2$ agents. We further improve this competitive ratio to $\sqrt{2} \approx 1.414$ by giving another efficient algorithm and provide a hard instance showing that no online algorithm can do better than $15/11 \approx 1.364$ competitive for $n = 2$. Moreover, we consider the case when items arrive in the order from the most costly to the least costly, and present an algorithm that is $5/3$-competitive for general number of agents. Finally, we consider small chores instances in which the cost of each item (to each agent) is bounded by some $\alpha < 1$ and demonstrate the existence of $(1+\alpha)$-competitive algorithm for general number of agents. For the case of two agents, we improve this competitive ratio to $\sqrt{\alpha^2 - 4\alpha + 5} + \alpha - 1$ for small chores instances (see full version). We summarize the upper and lower bounds on

[1] In full version, we provide some justifications for this assumption, showing that without this assumption (1) for the case of goods, the competitive ratio is arbitrarily bad; (2) for the case of chores, the problem becomes strictly harder.

Table 1. Summary of results, where Lower and Upper stand for Lower Bound and Upper Bound, respectively.

	Goods		Chores	
	Lower	Upper	Lower	Upper
$n \geq 3$	0	0	1.585 (Albers & Hellwig, 2012)	$2 - 1/n$
$n = 2$	0.5	0.5	$15/11$	$\sqrt{2}$

the competitive ratios in Table 1).

1.2. Other Related Works

In the traditional offline fair allocation problem, it has been shown that MMS allocations are not guaranteed to exist for goods (by Kurokawa et al. (2018)) and for chores (by Aziz et al. (2017)). Hence, many research focus on the computation of approximately MMS allocations, e.g., for goods (Kurokawa et al., 2018; Ghodsi et al., 2021; Garg & Taki, 2021) and chores (Aziz et al., 2017; Barman & Murthy, 2017; Huang & Lu, 2021). The state-of-the-art approximation ratio is $(\frac{3}{4} + \min\{\frac{1}{36}, \frac{3}{16n-4}\})$ for goods by Akrami et al. (2023) and $13/11$ for chores by Huang & Segal-Halevi (2023). Recently, Feige et al. (2021) show that no algorithm can achieve approximation ratios larger than $39/40$ for goods and smaller than $44/43$ for chores.

Similar to the literature on online scheduling, classic models for online allocation problems often focus on optimizing a global objective (Banerjee et al., 2022; Barman et al., 2022; Kawase & Sumita, 2022; Kellerer et al., 2015). Recently, inspired by many real-work applications, e.g., the allocation of food to charities in the food bank problem (Walsh, 2014; 2015; Aleksandrov et al., 2015), and the allocation of tasks to workers in scheduling problems (Kellerer et al., 1997; Cheng et al., 2005), some researchers turn to study the online fair allocation problem. In contrast to the rich literature on this problem for divisible items (Kash et al., 2014; Li et al., 2018; Bogomolnaia et al., 2022), the case of indivisible items is much less well-studied. To name a few, for allocating indivisible goods, Benade et al. (2018) propose algorithms for minimizing the expected maximum envy among agents; He et al. (2019) study the problem of minimizing the number of reallocations to ensure an EF1 allocation; Zeng & Psomas (2020) study the tradeoff between fairness and efficiency when the values of items are randomly drawn from a distribution.

For a more comprehensive review of other works on fair allocation and online fair allocation problems, please refer to the survey by Amanatidis et al. (2022) and Aleksandrov & Walsh (2020), respectively.

2. Preliminaries

We consider how to fairly allocate a set of m indivisible goods (or chores) M to a group of n agents N, where items M arrive online in an arbitrary order and agents N are offline. When items are goods, each agent $i \in N$ has a value $v_i(e) \geq 0$ on each item $e \in M$. That is, agent $i \in N$ has an additive valuation function $v_i : 2^M \to \mathbb{R}^+ \cup \{0\}$ that assigns a positive value $v_i(S) = \sum_{e \in S} v_i(e)$ to any subset of items $S \subseteq M$, and the agents would like to maximize their values. When items are chores, we use $c_i(e) \geq 0$ to denote the cost agent i has on item e, where c_i is the cost function. We have $c_i(S) = \sum_{e \in S} c_i(e)$ for all $S \subseteq E$, and the agents would like to minimize their costs. We assume that the valuation and cost functions are normalized, i.e., $v_i(M) = c_i(M) = n$ for all $i \in N$. An allocation is represented by an n-partition $\mathbf{X} = (X_1, \cdots, X_n)$ of the items, where $X_i \cap X_j = \emptyset$ for all $i \neq j$ and $\cup_{i \in N} X_i = M$. In allocation $\mathbf{X}$, agent $i \in N$ receives bundle X_i. Given any set $X \subseteq M$ and $e \in M$, we use $X + e$ and $X - e$ to denote $X \cup \{e\}$ and $X \setminus \{e\}$, respectively.

Definition 2.1 (MMS for Goods)**.** Let $\Pi(M)$ be the set of all n-partition of M. For the allocation of goods, for all agent $i \in N$, her maximin share (MMS) is defined as:

$$\mathsf{MMS}_i = \max_{X \in \Pi(M)} \min_{j \in N} \{v_i(X_j)\}.$$

For any $\alpha \in [0, 1]$, allocation $\mathbf{X}$ is α-approximate maximin share fair (α-MMS) if $v_i(X_i) \geq \alpha \cdot \mathsf{MMS}_i$ holds for all $i \in N$. When $\alpha = 1$, the allocation $\mathbf{X}$ is MMS.

Definition 2.2 (MMS for Chores)**.** Let $\Pi(M)$ be the set of all n-partition of M. For the allocation of chores, for all agent $i \in N$, her maximin share (MMS) is defined as:

$$\mathsf{MMS}_i = \min_{X \in \Pi(M)} \max_{j \in N} \{c_i(X_j)\}.$$

For any $\alpha \geq 1$, allocation $\mathbf{X}$ is α-approximate maximin share fair (α-MMS) if $c_i(X_i) \leq \alpha \cdot \mathsf{MMS}_i$ holds for any $i \in N$. When $\alpha = 1$, the allocation $\mathbf{X}$ is MMS.

Online Setting. We use $e_1, e_2, \ldots, e_m$ to index the items M in the order they arrive. We consider the adversarial setting in which the adversary designs the instance and decides the arrival order of items. Moreover, since our algorithms are deterministic, the adversary can be adaptive, that is, the adversary can design the value or cost of the online item depending on the previous decisions by the algorithm. The algorithm does not know m (the number of items), but knows the number of agents n and that $v_i(M) = n$ (for goods) or $c_i(M) = n$ (for chores). The value $v_i(e_j)$

(resp. cost $c_i(e_j)$) of item $e_j \in M$ is revealed for all $i \in N$ upon the arrival of e_j. Then the online algorithm must make an irrevocable decision on which agent $i \in N$ this item e_j should be assigned to. In other words, no reallocations of items are allowed. The performance of the algorithm is measured by the competitive ratio, which is the worst approximation guarantee (w.r.t. MMS) of the final allocation $\mathbf{X}$ over all online instances.

3. Allocation of Goods

We first consider the allocation of goods and provide upper and lower bounds for the competitive ratio of online algorithms for approximating MMS allocations. Recall that for the allocation of goods, the larger the competitive ratio the better, and thus upper bound corresponds to impossibility results while the lower bound corresponds to algorithmic results. As introduced, when agents have identical valuation functions, optimal competitive ratios $1/(n-1)$ for $n \geq 3$ (Tan & Wu, 2007) and $2/3$ for $n = 2$ (Kellerer et al., 1997) have been proved. In this section, we focus on the case when agents have general additive valuation functions. We first show that no online algorithm can guarantee a competitive ratio larger than 0, even for $n = 3$ agents. Then we propose our 0.5-competitive algorithm for the two-agent case.

Note that for the allocation of goods, we have $\mathsf{MMS}_i \leq (1/n) \cdot v_i(M) = 1$. Therefore as long as an agent receives a bundle with $v_i(X_i) \geq \gamma$, the allocation must be at least γ-MMS to her.

3.1. Upper Bound of Approximation Ratio

We show in this subsection that when there are at least 3 agents, the problem of approximating MMS allocations online does not admit any competitive algorithm.

Theorem 3.1. *No online algorithm has a competitive ratio strictly larger than* 0 *for approximating MMS allocations for goods, even when* $n = 3$.

Proof. We first consider the case of $n = 3$ agents and provide a collection of instances showing that no online algorithm can guarantee a competitive ratio strictly larger than 0 on these instances. The case when $n \geq 4$ can be proved in a very similar way, and we defer the proof to supplementary.

For the sake of contradiction, suppose there exists a γ-competitive algorithm for approximating MMS allocation for $n = 3$ agents, where $\gamma \in (0, 1]$. Let $r > 1/\gamma$ be a sufficiently large integer and $\epsilon > 0$ be sufficiently small such that $r^3\epsilon < \gamma$. We construct a collection of instances and show that for the allocation returned by the algorithm for at least one of these instances, at least one agent $i \in N$ is allocated a bundle X_i with $v_i(X_i) < (1/r) \cdot \mathsf{MMS}_i$. Note that since the allocation is deterministic, we can construct an instance gradually depending on how the previous items are allocated.

To begin with, let the first item be e_1 with $v_1(e_1) = v_2(e_1) = v_3(e_1) = \epsilon$ and assume w.l.o.g. that agent 1 receives it. Then let the second item be e_2 with

$$v_1(e_2) = r^2\epsilon, v_2(e_2) = v_3(e_2) = \epsilon.$$

Since the online algorithm has competitive ratio $\gamma > 1/r$, item e_2 can not be assigned to agent 1. This is because otherwise for the instance with only three items, where $v_1(e_3) = 3 - \epsilon - r^2\epsilon$ and $v_2(e_3) = v_3(e_3) = 3 - 2\epsilon$, there must exists an agent (in $\{2, 3\}$) that receives no item, which leads to a 0-MMS allocation. Since $v_2(e_1) = v_2(e_2) = v_3(e_1) = v_3(e_2)$, we can assume w.l.o.g. that agent 2 receives item e_2. Let e_3 be such that

$$v_1(e_3) = r\epsilon, v_2(e_3) = r^2\epsilon, v_3(e_3) = \epsilon.$$

We first show that e_3 can not be assigned to agent 2. Assume otherwise, i.e., $e_3 \in X_2$. Then for the instance with four items, where the last item e_4 has

$$v_1(e_4) = 3 - (\epsilon + r\epsilon + r^2\epsilon),$$
$$v_2(e_4) = 3 - (2\epsilon + r^2\epsilon), \ v_3(e_4) = 3 - 3\epsilon,$$

either $X_1 = \{e_1\}$ or $X_3 = \emptyset$ in the final allocation. Since $\mathsf{MMS}_1 = \epsilon + r\epsilon$ and $\mathsf{MMS}_3 > 0$, in both cases the competitive ratio is strictly smaller than γ. Hence every γ-competitive algorithm must allocate item e_3 to either agent 1 or 3. Depending on which agent receives item e_3, we construct two different instances.

For the case when agent 1 receives item e_3, we construct the instance with five items as in Table 2.

Table 2. Instance with $m = 5$ when e_3 assigned to agent 1.

	e_1	e_2	e_3	e_4	e_5
1	$\boxed{\epsilon}$	$r^2\epsilon$	$\boxed{r\epsilon}$	$r^3\epsilon$	$3 - (r^3\epsilon + r^2\epsilon + r\epsilon + \epsilon)$
2	ϵ	$\boxed{\epsilon}$	$r^2\epsilon$	$r\epsilon$	$3 - (r^2\epsilon + r\epsilon + 2\epsilon)$
3	ϵ	ϵ	ϵ	ϵ	$3 - 4\epsilon$

Let X'_i be the bundle agent i holds at the moment, for each $i \in N$. We have $\mathsf{MMS}_1 = r^2\epsilon + r\epsilon + \epsilon$, $\mathsf{MMS}_2 = r\epsilon + 2\epsilon$, $\mathsf{MMS}_3 = 2\epsilon$ and $v_1(X'_1) = v_1(e_1 + e_3) = r\epsilon + \epsilon, v_2(X'_2) = v_2(e_2) = \epsilon$, $v_3(X'_3) = 0$. For all $i \in N$, $v_i(X'_i) < 1/r \cdot \mathsf{MMS}_i$. Observed that there must exist at least one agent $i \in N$ that does not receive any item in $\{e_4, e_5\}$ in the final allocation. In other words, we have $X_i = X'_i$, which leads to a contradiction that the algorithm computes γ-MMS allocations.

Next, we consider the case when agent 3 receives item e_3. Let the next item e_4 be such that

$$v_1(e_4) = \epsilon, v_2(e_4) = r\epsilon, v_3(e_4) = r^2\epsilon.$$

We argue that the algorithm can not allocate item e_4 to agent 3. Consider the following instance (as shown in Table 3).

Table 3. Instance showing e_4 cannot be assigned to agent 3.

	e_1	e_2	e_3	e_4	e_5
1	$\boxed{\epsilon}$	$r^2\epsilon$	$r\epsilon$	ϵ	$3-(r^2\epsilon+r\epsilon+2\epsilon)$
2	ϵ	$\boxed{\epsilon}$	$r^2\epsilon$	$r\epsilon$	$3-(r^2\epsilon+r\epsilon+2\epsilon)$
3	ϵ	ϵ	$\boxed{\epsilon}$	$r^2\epsilon$	$3-(r^2\epsilon+3\epsilon)$

Note that for this instance we have

$$\mathsf{MMS}_1 = \mathsf{MMS}_2 = r\epsilon + 2\epsilon,$$

while at the moment we have $v_1(X_1') = v_2(X_2') = \epsilon$. Therefore the agent $i \in \{1, 2\}$ that does not receive item e_5 in the final allocation is not γ-MMS. Therefore we know that the algorithm must allocate item e_4 to agent 1 or 2, for which case we construct the following instance with six items (as shown in Table 4).

Note that for this instance we have

$$\mathsf{MMS}_1 = r^2\epsilon + 2\epsilon,\ \mathsf{MMS}_2 = r^2\epsilon + r\epsilon + 2\epsilon,$$
$$\mathsf{MMS}_3 = r^2\epsilon + \epsilon.$$

On the other hand, (no matter which agent receives item e_4) we have

$$v_1(X_1') \le 2\epsilon,\ v_2(X_2') \le r\epsilon + \epsilon,\ v_3(X_3') = \epsilon.$$

Since only items $\{e_5, e_6\}$ are not allocated, for the agent $i \in N$ that does not receive any item in $\{e_5, e_6\}$ in the final allocation, the allocation is not γ-MMS to her.

In summary, no online algorithm is γ-competitive for three agents. Extending this result to $n \ge 4$ agents is almost straightforward, and we refer the proof to the full version. □

3.2. Two Agents

In this section, we consider the case with two agents and show that a 0.5-competitive algorithm exists, and is indeed optimal. A natural algorithmic idea for two agents is to allocate each item to the agent who values the items more (in a greedy manner) and stop allocating any further item to an agent once its total value exceeds 0.5. Unfortunately, via the following instance (see Table 5) we show that the allocation returned by this greedy algorithm can be arbitrarily bad.

In this instance, both items e_1, e_2 are allocated to agent 2 since she values them more, and her value does not exceed 0.5 at the moment of allocation. However, the allocation is far from being MMS fair to agent 1, since $\mathsf{MMS}_1 = 0.5 + \epsilon$ and $v_1(X_1) = 3\epsilon$. The algorithm fails because when there exists an item with very large value, e.g., item e_2 in the instance, the greedy algorithm may allocate this item to the agent with value close to 0.5, which results in a significantly unfair allocation. To fix this issue, we propose the following algorithm that equips the greedy algorithm with a special handling of large items.

Algorithm for Two Agents. We call an item e *large* to agent i if $v_i(e) \ge 0.5$. For each online item e, if it is large to both agents, we assign e to the agent with smaller $v_i(X_i)$ at the moment and allocate all future items to the other agent. Otherwise, we allocate e to the agent i with larger $v_i(e)$. Once we have $v_i(X_i) \ge 1/2$ for some agent $i \in N$, we allocate all future items to the other agent $j \neq i$ (refer to Algorithm 1). Throughout the algorithm (and all later algorithms), we break ties arbitrarily but consistently, e.g., using the id of agents.

Algorithm 1 Algorithm-for-2-Agents-for-Goods

```
Initialize: $X_1, X_2 \leftarrow \emptyset$ and $A \leftarrow \{1,2\}$ be the active agents
for each online item $e \in M$ do
  if $|A| = 1$ then
    $X_i \leftarrow X_i + e$, where $i \in A$
  else if $v_1(e) \ge 1/2$ and $v_2(e) \ge 1/2$ then
    $i \leftarrow \text{argmin}_{j\in N}\{v_j(X_j)\}$, $X_i \leftarrow X_i + e$
    turn agent $i$ into inactive: $A \leftarrow A \setminus \{i\}$
  else
    $i \leftarrow \text{argmax}_{j\in N}\{v_j(e)\}$, $X_i \leftarrow X_i + e$
    if $v_i(X_i) \ge 1/2$ then
      turn agent $i$ into inactive: $A \leftarrow A \setminus \{i\}$
    end if
  end if
end for
Output: $\mathbf{X} = (X_1, X_2)$
```

Theorem 3.2. *For $n = 2$ agents, Algorithm 1 computes an 0.5-MMS allocation in $O(m)$ time.*

Proof. Let $\mathbf{X} = (X_1, X_2)$ be the final allocation. We first show that if no item $e \in M$ is large to both agents, then $\mathbf{X}$ is 1/2-MMS. Since each item e is allocated to the agent with larger $v_i(e)$, we have $v_i(X_i) \ge v_j(X_i)$ for all $i \in N$ and $j \neq i$. Therefore in the final allocation, we have

$$v_1(X_1) + v_2(X_2) \ge v_1(M) = 2.$$

Hence at least one of the agents, says agent i, will be turned inactive by the algorithm. Let e be the last item agent i receives. Then we have $v_i(e) \ge v_j(e)$ for $j \neq i$ and $v_i(X_i - e) < 0.5$ by the design of the algorithm. Hence we have

$$v_j(X_i) = v_j(X_i - e) + v_j(e) < 1/2 + 1/2 = 1,$$

where the inequality holds because $v_j(X_i - e) \le v_i(X_i - e)$ (since each item is allocated to the agent that values it more),

Table 4. Suppose e_4 is allocated to one of the agents in $\{1, 2\}$.

	e_1	e_2	e_3	e_4	e_5	e_6
1	$\boxed{\epsilon}$	$r^2\epsilon$	$r\epsilon$	ϵ	$r^2\epsilon$	$3-(2r^2\epsilon+r\epsilon+2\epsilon)$
2	ϵ	$\boxed{\epsilon}$	$r^2\epsilon$	$r\epsilon$	$r^3\epsilon$	$3-(r^3\epsilon+r^2\epsilon+r\epsilon+2\epsilon)$
3	ϵ	ϵ	$\boxed{\epsilon}$	$r^2\epsilon$	$r^2\epsilon$	$3-(2r^2\epsilon+3\epsilon)$

Table 5. Hard instance for the greedy algorithm, where $\epsilon > 0$ is arbitrarily small.

	e_1	e_2	e_3
1	$0.5-2\epsilon$	$1.5-\epsilon$	$\boxed{3\epsilon}$
2	$\boxed{0.5-\epsilon}$	$\boxed{1.5}$	ϵ

and that item e is not large to agent j. Hence we have $v_j(X_j) = v_j(M) - v_j(X_i) > 1$. Recall that $v_i(X_i) \geq 1/2$. Hence the allocation $\mathbf{X}$ is a $1/2$-MMS.

Next, we argue that if there exists an item $e \in M$ large to both agents, then $\mathbf{X}$ is also $1/2$-MMS. Let item e be the first item large to both agents and suppose that it is allocated to agent i. Then we have $v_i(X_i) \geq v_i(e) \geq 1/2 \geq 1/2 \cdot \mathsf{MMS}_i$. Next we show that $v_j(X_j) \geq 1/2 \cdot \mathsf{MMS}_j$ for the other agent $j \neq i$. Let X'_1 (resp. X'_2) be the bundle agent 1 (resp. 2) holds before item e is allocated. By the design of the algorithm we have $v_i(X'_i) \leq v_j(X'_j)$. Since all items that arrive before e are allocated greedily, we have $v_j(X'_j) \geq v_i(X'_i) \geq v_j(X'_i)$. Since all items that arrive after e are allocated to agent j, we have

$$\begin{aligned} v_j(X_j) &\geq \frac{1}{2} \cdot (v_j(X_j) + v_j(X'_i)) \\ &= \frac{1}{2} \cdot v_j(M-e) \geq \frac{1}{2} \cdot \mathsf{MMS}_j, \end{aligned}$$

where the last inequality holds because in any allocation the bundle not containing item e has value at most $v_j(M-e)$, which implies $\mathsf{MMS}_j \leq v_j(M-e)$. Since the allocation of each item $e \in M$ takes $O(1)$ time, the algorithm runs in $O(m)$ time. □

Next, we show that no online algorithm can achieve a competitive ratio strictly better than 0.5. Therefore, our algorithm is optimal.

Theorem 3.3. *For $n = 2$ agents, no online algorithm has a competitive ratio strictly larger than* 0.5*.*

Proof. Assume the contrary that there exists an online algorithm with competitive ratio $0.5 + \delta$ for some $\delta > 0$. We first prove that this algorithm must maintain the following property throughout the execution of the whole algorithm, which (roughly speaking) enforces the algorithm to maintain an "uneven" allocation at all times.

Claim 3.1. *Throughout the execution of a $(0.5 + \delta)$-competitive algorithm, if $v_1(X_1 + X_2) \leq 1$ and $v_2(X_1 + X_2) \leq 1$, then there must exists $i \in N$ such that $v_i(X_i) > (1 + \delta) \cdot v_i(X_j)$.*

Following Claim 3.1 we construct an instance (see Table 6), and show that the algorithm must allocate all items $e_1, \ldots, e_k$ to agent 1 (we can assume w.l.o.g. that agent 1 receives the first item).

Specifically, the first item e_1 has $v_1(e_1) = v_2(e_1) = \epsilon$ and we can assume w.l.o.g that agent 1 receives it. Then we construct items $e_2, \cdots, e_k$ such that for all $1 < i \leq k$, we have

$$v_1(e_i) = \frac{(2+\delta)^{i-2} \cdot \epsilon}{(1+\delta)^{i-1}},$$

$$v_2(e_i) = \begin{cases} (2+\delta)^{i-2} \cdot (1+\delta) \cdot \epsilon, & \text{if } i \leq l \\ (2+\delta)^{l-2} \cdot (1+\delta) \cdot \epsilon, & \text{if } i > l. \end{cases}$$

Observe that for all $1 < i \leq k$, we have

$$\begin{aligned} \sum_{j=1}^{i-1} v_1(e_j) &= \epsilon + \frac{\epsilon}{1+\delta} \cdot \sum_{j=2}^{i-1} \left(\frac{2+\delta}{1+\delta}\right)^{j-2} \\ &= \epsilon \cdot \left(\frac{2+\delta}{1+\delta}\right)^{i-2} \\ &= (1+\delta) \cdot v_1(e_i). \end{aligned} \tag{1}$$

For all $1 < i \leq l$, we have

$$\begin{aligned} \sum_{j=1}^{i-1} v_2(e_j) &= \epsilon + \epsilon \cdot (1+\delta) \cdot \sum_{j=2}^{i-1} (2+\delta)^{j-2} \\ &= \epsilon \cdot (2+\delta)^{i-2} = \frac{v_2(e_i)}{1+\delta}. \end{aligned} \tag{2}$$

In other words, item e_i has value roughly the same as all items before i combined, for both agents. However, the value grows slightly faster in v_2 than in v_1. This is also reflected by the definition of $v_1(e_i)$ and $v_2(e_i)$: observe that for all $1 < i \leq l$ we have $\frac{v_2(e_i)}{v_1(e_i)} = (1+\delta)^i$, i.e., the difference in value under the two valuation functions grows exponentially in i. Note that all items $e_l, e_{l+1}, \ldots, e_k$ have the same value $(2+\delta)^{l-2} \cdot (1+\delta) \cdot \epsilon$ to agent 2. By picking $\epsilon = \frac{0.1}{(1+\delta)(2+\delta)^{l-2}}$ we can ensure that

$$\begin{aligned} v_2(e_l) &= v_2(e_{l+1}) = \cdots = v_2(e_k) \\ &= (2+\delta)^{l-2} \cdot (1+\delta) \cdot \epsilon = 0.1. \end{aligned}$$

	e_1	$\cdots$	e_l	$\cdots$	e_k	e_{k+1}
1	ϵ	$\cdots$	$\frac{(2+\delta)^{l-2}\cdot\epsilon}{(1+\delta)^{l-1}}$	$\cdots$	$\frac{(2+\delta)^{k-2}\cdot\epsilon}{(1+\delta)^{k-1}}$	>1.9
2	ϵ	$\cdots$	$(2+\delta)^{l-2}(1+\delta)\epsilon$	$\cdots$	$(2+\delta)^{l-2}(1+\delta)\epsilon$	<0.1

Table 6. Hard instance for allocation of goods for two agents, where $\epsilon > 0$ is arbitrarily small.

Finally, let e_{k+1} be the last item with $v_1(X_1+X_2+e_{k+1}) = v_2(X_1+X_2+e_{k+1}) = 2$. By setting $k = l + 18$, we have (where the third equality follows from (2) and $v_2(e_l) = 0.1$)

$$\begin{aligned} v_2(e_{k+1}) &= 2 - \sum_{i=1}^{k} v_2(e_i) \\ &= 2 - \sum_{i=1}^{l-1} v_2(e_i) - (k-l+1)\cdot v_2(e_l) \\ &= 2 - \frac{0.1}{1+\delta} - 1.9 = \frac{0.1\cdot\delta}{1+\delta}. \end{aligned}$$

By setting l to be sufficiently large (which also defines ϵ and k), we can ensure that

$$\begin{aligned} v_1(e_{k+1}) &= 2 - \sum_{i=1}^{k} v_1(e_i) = 2 - \epsilon\cdot\left(\frac{2+\delta}{1+\delta}\right)^{k-1} \\ &= 2 - \frac{0.1\cdot(2+\delta)^{19}}{(1+\delta)^{l+18}} > 1.9. \end{aligned}$$

We argue that the algorithm must allocate the first k items to agent 1.

Claim 3.2. *The algorithm with competitive ratio $0.5+\delta$ must assign all items $e_2, \cdots, e_k$ to agent 1.*

Given Claim 3.2, we know that when item e_{k+1} arrives, agent 2 is not allocated any item, which leads to $v_2(X_2) < 0.1$ in the final allocation. Since $\mathsf{MMS}_2 = 1$, the allocation is obviously not $(0.5+\delta)$-MMS to agent 2, which is also a contradiction. $\square$

4. Allocation of Chores

In this section, we consider the allocation of chores. Recall that for the allocation of chores, each agent $i \in N$ has a cost $c_i(e) \geq 0$ on item $e \in M$, and the competitive ratios are at least 1 (thus the smaller the better). Also recall that when agents have identical cost functions, optimal competitive ratios 1.585 (Kellerer et al., 2015) and 4/3 (Kellerer et al., 1997) have been proved for $n \geq 3$ and $n = 2$, respectively. We focus on the case when agents have general additive cost functions. In contrast to the allocation of goods, we show that MMS allocation of chores admits constant competitive algorithms even for general number of agents. For $n \geq 3$ agents, we propose a $(2-1/n)$-competitive algorithm using a similar idea of greedy allocation as in (Li et al., 2022); for $n = 2$ agents, we improve the competitive ratio to $\sqrt{2}$ and show that the competitive ratio is at least $15/11$ for any online algorithm. Note that for the allocation of chores, we have $\mathsf{MMS}_i \geq (1/n)\cdot c_i(M) = 1$. Therefore as long as an agent receives a bundle with cost $c_i(X_i) \leq \alpha$, the allocation must be α-MMS to her. Moreover, we have $\mathsf{MMS}_i \geq \max_{e\in M}\{c_i(e)\}$, because in any allocation the bundle with maximum cost should have cost at least as large as that of the most costly item.

4.1. Online Approximation Algorithm

We first consider the general case with n agents and present our algorithm that computes a $(2-1/n)$-MMS allocations in $O(mn)$ time. The algorithm follows a similar idea as we have used in the previous section (refer to Algorithm 2): (1) each online item is greedily allocated to the active agent that has minimum cost on the item; (2) once an agent receives a collection of items of large cost ($\geq 1-1/n$ in our algorithm), we inactivate this agent. Initially, all agents are active, and if at some moment only one agent is active, then all future items will be allocated to this agent.

Algorithm 2 Algorithm-for-n-Agents-for-Chores

```
Initialize: $A \leftarrow N$ and for any $i \in N$, $X_i \leftarrow \emptyset$
for each online item $e \in M$ do
  if $|A| = 1$ then
    $X_i \leftarrow X_i + e$, where $i \in A$
  else
    $i \leftarrow \mathrm{argmin}_{j\in A}\{c_j(e)\}$;
    $X_i \leftarrow X_i + e$
    if $c_i(X_i) \geq 1 - 1/n$ then
      turn agent $i$ into inactive: $A \leftarrow A \setminus \{i\}$
    end if
  end if
end for
Output: $\mathbf{X} = (X_1, X_2, \cdots, X_n)$
```

Theorem 4.1. *For $n \geq 2$ agents, Algorithm 2 computes a $(2-\frac{1}{n})$-MMS allocation in $O(mn)$ time.*

Proof. Note that throughout the execution of the algorithm, if $|A| \geq 2$, then all active agents $i \in A$ has $c_i(X_i) < 1 - 1/n$. Therefore, for any agent i that is inactivated, let e_i be the last item agent i receives, then we have

$$c_i(X_i) < 1 - 1/n + c_i(e_i) \leq (2-1/n)\cdot\mathsf{MMS}_i,$$

where the inequality follows from the fact that $\mathsf{MMS}_i \geq \max_{e\in M}\{c_i(e)\}$ and $\mathsf{MMS}_i \geq 1$ for the allocation of chores. Hence the final allocation is $(2-1/n)$-MMS to all inactive agents, and it remains to consider the case when $|A|=1$ at the end of the algorithm.

Let i be the only active agent to which the last item is allocated. By the design of the algorithm, for each $j \neq i$ (that is already inactivated), we have $c_j(X_j) \geq 1-1/n$. Moreover, by the greedy allocation of the algorithm, for all $e \in X_j$ we have $c_j(e) \leq c_i(e)$ (because both agents i and j are active when e arrives). Hence we have $c_i(X_j) \geq c_j(X_j)$, which implies

$$\begin{aligned} c_i(X_i) &= c_i(M) - \sum_{j\neq i} c_i(X_j) \\ &\leq n-(n-1)\cdot(1-1/n) \\ &= 2-1/n \leq (2-1/n)\cdot \mathsf{MMS}_i. \end{aligned}$$

Hence, the allocation is also $(2-1/n)$-MMS to agent i. Since the allocation of each item takes $O(n)$ time, the algorithm returns a $(2-1/n)$-MMS allocation in $O(nm)$ time. □

4.2. Two Agents

The above result gives a 1.5-competitive algorithm for $n=2$ agents. In this section, we improve the ratio to $\sqrt{2}$. We complement our algorithmic result with hard instances showing that no online algorithm can do better than $15/11$-competitive. The key observation towards this improvement is to mimic the bin-packing algorithms (Aziz et al., 2017; Huang & Lu, 2021) for approximating MMS for chores. In particular, as long as the costs of the two agents do not differ by a factor larger than $\sqrt{2}$, we deviate from the greedy allocation and treat the item as equally costly to the two agents, and allocate the item to a designated agent (agent 1 in our algorithm). In addition, to ensure a bounded competitive ratio, we dynamically update a lower bound α_i for MMS_i, for both $i \in \{1,2\}$. Our algorithm makes sure that each allocation of item does not result in $c_i(X_i) \geq \sqrt{2}\cdot\alpha_i$ for both $i\in\{1,2\}$.

Algorithm for Two Agents. Throughout the execution of the algorithm, we maintain that $\alpha_i = \max\{1, \max_{e:\text{arrived}}\{c_i(e)\}\}$. For each online item e, we first identify the agents $A=\{i\in N : c_i(X_i+e)\leq\sqrt{2}\cdot\alpha_i\}$ that can receive the item e without violating the competitive ratio $\sqrt{2}$. If $|A|=1$ then we allocate e to the only agent in A; otherwise if $c_1(e)\leq\sqrt{2}\cdot c_2(e)$, we allocate item e to agent 1; otherwise we allocate item e to agent 2 (refer to Algorithm 3).

Theorem 4.2. *For $n=2$ agents, Algorithm 3 computes a $\sqrt{2}$-MMS allocation in $O(m)$ time.*

Algorithm 3 Algorithm-for-Two-Agents-for-Chores

```
Initialize: $X_1 \leftarrow \emptyset$, $X_2 \leftarrow \emptyset$, $\alpha_1 \leftarrow 1$ and $\alpha_2 \leftarrow 1$
for each online item $e \in M$ do
    update $\alpha_1 \leftarrow \max\{\alpha_1, c_1(e)\}$
    update $\alpha_2 \leftarrow \max\{\alpha_2, c_2(e)\}$
    $A \leftarrow \{i \in N : c_i(X_i+e) \leq \sqrt{2}\cdot\alpha_i\}$
    if $|A| = 1$ then
        $X_i \leftarrow X_i + e$, where $i \in A$
    else if $c_1(e) \leq \sqrt{2}\cdot c_2(e)$ then
        $X_1 \leftarrow X_1 + e$
    else
        $X_2 \leftarrow X_2 + e$
    end if
end for
Output: $\mathbf{X} = (X_1, X_2)$
```

Proof. Since we always have $\mathsf{MMS}_i \geq \alpha_i$, to show that the returned allocation is $\sqrt{2}$-MMS, it suffices to show that when each item e arrives, the set A is not empty. Because when $A \neq \emptyset$, our algorithm will allocate e to some agent $i \in A$, which ensures that $c_i(X_i+e)\leq\sqrt{2}\cdot\alpha_i\leq\sqrt{2}\cdot\mathsf{MMS}_i$. For the sake of contradiction, we assume that when some online item e^* arrives we have $A=\emptyset$. That is, we have $c_i(X_i+e^*)>\sqrt{2}\cdot\alpha_i$ for both $i\in\{1,2\}$, where X_i is the bundle agent i holds when e arrives. We claim that under this situation, we have $c_1(e')>\sqrt{2}\cdot c_2(e')$ for all item $e'\in X_2$.

Claim 4.1. *For all $e' \in X_2$, $c_1(e') > \sqrt{2}\cdot c_2(e')$.*

Proof. Suppose otherwise and let e_1 be the first item assigned to X_2 with $c_1(e_1)\leq\sqrt{2}\cdot c_2(e_1)$. Let X_1' be the bundle agent 1 holds when e_1 arrives. By the design of the algorithm we must have $c_1(X_1'+e_1)>\sqrt{2}\cdot\alpha_1\geq\sqrt{2}$, because otherwise e_1 will be allocated to agent 1. Recall that we assumed $c_1(X_1+e^*)>\sqrt{2}\cdot\alpha_1\geq\sqrt{2}$ for some item e^* that is not allocated in the final allocation, which implies

$$c_1(X_2)\leq c_1(M)-c_1(X_1+e^*)<2-\sqrt{2}.$$

Therefore we have $c_1(e_1)\leq c_1(X_2)<2-\sqrt{2}$ (because $e_1\in X_2$), which gives

$$c_1(X_1')=c_1(X_1'+e_1)-c_1(e_1)>2\sqrt{2}-2.$$

On the other hand, since $e_1\in X_2$ is the first item with $c_1(e_1)\leq\sqrt{2}\cdot c_2(e_1)$, we have

$$c_2(X_1')\geq c_1(X_1')/\sqrt{2}>2-\sqrt{2},$$

which implies $c_2(X_2+e^*)\leq c_2(M-X_1')<\sqrt{2}$, and it is a contradiction with our previous assumption that $c_2(X_2+e^*)>\sqrt{2}\cdot\alpha_2$ for some item e^* (that arrives after e_1). □

Note that $c_1(X_2) \le c_1(M) - c_1(X_1 + e^*) < 2 - \sqrt{2}$. By Claim 4.1,

$$c_2(X_2) < c_1(X_2)/\sqrt{2} < \sqrt{2} - 1.$$

Recall that $\alpha_2 \ge \max\{1, c_2(e^*)\}$, we have $c_2(X_2 + e^*) \le \sqrt{2} - 1 + c_2(e^*) \le \sqrt{2} \cdot \alpha_2$, which is a contradiction. Finally, since the allocation of each item takes $O(1)$ time, the algorithm executes in $O(m)$ time. □

Next, we present a collection of instances and show that no online algorithm can have a competitive ratio smaller than $15/11$.

Theorem 4.3. *For $n = 2$ agents, no online algorithm has a competitive ratio smaller than* $15/11$.

Proof. Assume the contrary and suppose there exists an online algorithm with a competitive ratio smaller than $15/11$. In the following, we construct some instances and show that the algorithm returns an allocation with an approximation ratio (w.r.t. to MMS) of at least $15/11$ on at least one of the instances, which is a contradiction.

Let the first item be e_1 with $c_1(e_1) = c_2(e_1) = 4/11$, and we assume w.l.o.g. that it is allocated to agent 1. Then we construct the second item e_2 that values $4/11$ to agent 1 and $3/11$ to agent 2. We argue that the algorithm with a competitive ratio smaller than $15/11$ can not assign e_2 to agent 1.

Table 7. Assume item e_2 is assigned to agent 1.

	e_1	e_2	e_3	e_4
1	$\boxed{4/11}$	$\boxed{4/11}$	$7/11$	$7/11$
2	$4/11$	$3/11$	$7/11$	$8/11$

Assume otherwise, i.e., item e_2 is also allocated to agent 1, and we consider the instance as shown in Table 7. Note that for this instance we have $\mathsf{MMS}_1 = \mathsf{MMS}_2 = 1$. However, no matter how e_3 and e_4 are allocated, there must be an agent with a total cost of at least $15/11$, which is a contradiction. Therefore the algorithm must allocate e_2 to agent 2. Then we construct the following instance with 7 items (see Table 8), and show that the algorithm must allocate items e_3, e_4, e_5 and e_6 to agent 1.

Table 8. Assume item e_2 is assigned to agent 2.

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
1	$\boxed{\frac{4}{11}}$	$\frac{4}{11}$	$\boxed{\frac{3}{11}}$	$\boxed{\frac{3}{11}}$	$\boxed{\frac{3}{11}}$	$\boxed{\frac{3}{11}}$	$\frac{2}{11}$
2	$\frac{4}{11}$	$\boxed{\frac{3}{11}}$	$\frac{1}{11}$	$\frac{1}{11}$	$\frac{1}{11}$	$\frac{1}{11}$	1

Assume otherwise and let e_i be the first item in $\{e_3, e_4, e_5, e_6\}$ allocated to agent 2. Note that right after the allocation we have $c_2(X_2) = 4/11$. Then we consider another instance in which the next item e_{i+1} has

$$c_1(e_{i+1}) = \begin{cases} 1, & \text{if } i = 3 \\ 8/11, & \text{if } i = 4 \\ 5/11, & \text{if } i = 5 \\ 2/11, & \text{if } i = 6 \end{cases} \quad \text{and} \quad c_2(e_{i+1}) = 1.$$

It can be verified that in all cases, $\mathsf{MMS}_1 = \mathsf{MMS}_2 = 1$ but whoever receives item e_{i+1} would have cost at least $15/11$, which is a contradiction. Hence we have $\{e_3, e_4, e_5, e_6\} \subseteq X_1$, which is also a contradiction because $c_1(X_1) \ge 16/11 > 15/11$ (see Table 8). □

5. Conclusion and Open Problems

In this paper, we study the problem of fairly allocating indivisible online items to a group of agents with general additive valuation functions. For the allocation of goods, we show that no algorithm can guarantee any non-zero competitive ratio for $n \ge 3$ agents and propose an optimal 0.5-competitive algorithm for two agents. For the allocation of chores, we propose a $(2 - 1/n)$-competitive algorithm for $n \ge 3$ agents, a $\sqrt{2}$-competitive algorithm for two agents, and show that no algorithm can do better than $15/11$-competitive for two agents. We also study the monotone instances for both goods and chores and improve the competitive ratios to 0.5 and $5/3$ for goods and chores, respectively. We further consider the small items instances such that all value/cost are not greater than α where $(1-\alpha)$-competitive algorithm and $(1+\alpha)$-competitive algorithm exist for goods and chores respectively and improve the competitive ratio to $\sqrt{\alpha^2 - 4\alpha + 5} + \alpha - 1$ when allocating small chores to two agents.

There are many open problems regarding the online approximation of MMS allocations. First, while we show that no competitive algorithm exists for the online allocation of goods and competitive algorithms exist for monotone instances, we are curious about a less restrictive condition under which competitive algorithms exist. Second, for the allocation of chores, the lower bound of 1.585 for general number of agents follows from the identical valuation case, and it remains unknown whether the general additive valuation case is strictly harder. It is also interesting to investigate the optimal competitive ratio (which is in $[1.585, 2)$) for general number of agents and that for the case of two agents (which is in $[1.364, 1.414]$). Finally, while we show that normalization of the valuation functions is necessary to achieve a bounded ratio for the allocation of goods, it remains unknown whether constant competitive ratios can be achieved for chores if the cost functions are not normalized.

References

Akrami, H., Garg, J., Sharma, E., and Taki, S. Simplification and improvement of MMS approximation. *CoRR*, abs/2303.16788, 2023.

Albers, S. and Hellwig, M. Semi-online scheduling revisited. *Theor. Comput. Sci.*, 443:1–9, 2012.

Aleksandrov, M. and Walsh, T. Online fair division: A survey. In *AAAI*, pp. 13557–13562. AAAI Press, 2020.

Aleksandrov, M., Aziz, H., Gaspers, S., and Walsh, T. Online fair division: Analysing a food bank problem. In *IJCAI*, pp. 2540–2546. AAAI Press, 2015.

Amanatidis, G., Aziz, H., Birmpas, G., Filos-Ratsikas, A., Li, B., Moulin, H., Voudouris, A. A., and Wu, X. Fair division of indivisible goods: A survey. *CoRR*, abs/2208.08782, 2022.

Angelelli, E., Speranza, M. G., and Tuza, Z. Semi on-line scheduling on three processors with known sum of the tasks. *Journal of Scheduling*, 10(4):263–269, 2007.

Aziz, H., Rauchecker, G., Schryen, G., and Walsh, T. Algorithms for max-min share fair allocation of indivisible chores. In *AAAI*, pp. 335–341. AAAI Press, 2017.

Aziz, H., Li, B., and Wu, X. Approximate and strategyproof maximin share allocation of chores with ordinal preferences. *Mathematical Programming*, pp. 1–27, 2022.

Backurs, A., Indyk, P., Onak, K., Schieber, B., Vakilian, A., and Wagner, T. Scalable fair clustering. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 405–413. PMLR, 2019.

Banerjee, S., Gkatzelis, V., Gorokh, A., and Jin, B. Online nash social welfare maximization with predictions. In *SODA*, pp. 1–19. SIAM, 2022.

Barman, S. and Murthy, S. K. K. Approximation algorithms for maximin fair division. In *EC*, pp. 647–664. ACM, 2017.

Barman, S., Khan, A., and Maiti, A. Universal and tight online algorithms for generalized-mean welfare. In *AAAI*, pp. 4793–4800. AAAI Press, 2022.

Benade, G., Kazachkov, A. M., Procaccia, A. D., and Psomas, C. How to make envy vanish over time. In *EC*, pp. 593–610. ACM, 2018.

Bogomolnaia, A., Moulin, H., and Sandomirskiy, F. On the fair division of a random object. *Manag. Sci.*, 68(2): 1174–1194, 2022.

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

Caragiannis, I., Kurokawa, D., Moulin, H., Procaccia, A. D., Shah, N., and Wang, J. The unreasonable fairness of maximum nash welfare. *ACM Trans. Economics and Comput.*, 7(3):12:1–12:32, 2019.

Chen, X., Fain, B., Lyu, L., and Munagala, K. Proportionally fair clustering. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1032–1041. PMLR, 2019.

Cheng, S. Multistage online maxmin allocation of indivisible entities. *Theor. Comput. Sci.*, 933:104–113, 2022.

Cheng, T. C. E., Kellerer, H., and Kotov, V. Semi-on-line multiprocessor scheduling with given total processing time. *Theor. Comput. Sci.*, 337(1-3):134–146, 2005.

Ebenlendr, T., Noga, J., Sgall, J., and Woeginger, G. J. A note on semi-online machine covering. In *WAOA*, volume 3879 of *Lecture Notes in Computer Science*, pp. 110–118. Springer, 2005.

Feige, U., Sapir, A., and Tauber, L. A tight negative example for MMS fair allocations. In *WINE*, volume 13112 of *Lecture Notes in Computer Science*, pp. 355–372. Springer, 2021.

Foley, D. Resource allocation and the public sector. *Yale Economic Essays*, pp. 45–98, 1967.

Garg, J. and Taki, S. An improved approximation algorithm for maximin shares. *Artif. Intell.*, 300:103547, 2021.

Ghodsi, M., Hajiaghayi, M. T., Seddighin, M., Seddighin, S., and Yami, H. Fair allocation of indivisible goods: Improvement. *Math. Oper. Res.*, 46(3):1038–1053, 2021.

Gkatzelis, V., Psomas, A., and Tan, X. Fair and efficient online allocations with normalized valuations. In *AAAI*, pp. 5440–5447. AAAI Press, 2021.

He, J., Procaccia, A. D., Psomas, A., and Zeng, D. Achieving a fairer future by changing the past. In *IJCAI*, pp. 343–349. ijcai.org, 2019.

Huang, X. and Lu, P. An algorithmic framework for approximating maximin share allocation of chores. In *EC*, pp. 630–631. ACM, 2021.

Huang, X. and Segal-Halevi, E. A reduction from chores allocation to job scheduling. *CoRR*, abs/2302.04581, 2023.

Kash, I. A., Procaccia, A. D., and Shah, N. No agent left behind: Dynamic fair division of multiple resources. *J. Artif. Intell. Res.*, 51:579–603, 2014.

Kawase, Y. and Sumita, H. Online max-min fair allocation. In *SAGT*, volume 13584 of *Lecture Notes in Computer Science*, pp. 526–543. Springer, 2022.

Kellerer, H., Kotov, V., Speranza, M. G., and Tuza, Z. Semi on-line algorithms for the partition problem. *Oper. Res. Lett.*, 21(5):235–242, 1997.

Kellerer, H., Kotov, V., and Gabay, M. An efficient algorithm for semi-online multiprocessor scheduling with given total processing time. *J. Sched.*, 18(6):623–630, 2015.

Kurokawa, D., Procaccia, A. D., and Wang, J. Fair enough: Guaranteeing approximate maximin shares. *J. ACM*, 65 (2):8:1–8:27, 2018.

Lee, K. and Lim, K. Semi-online scheduling problems on a small number of machines. *J. Sched.*, 16(5):461–477, 2013.

Li, B., Li, W., and Li, Y. Dynamic fair division problem with general valuations. In *IJCAI*, pp. 375–381. ijcai.org, 2018.

Li, B., Li, L., Sun, A., Wang, C., and Wang, Y. Approximate group fairness for clustering. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6381–6391. PMLR, 2021.

Li, B., Li, Y., and Wu, X. Almost (weighted) proportional allocations for indivisible chores. In *WWW*, pp. 122–131. ACM, 2022.

Lipton, R. J., Markakis, E., Mossel, E., and Saberi, A. On approximately fair allocations of indivisible goods. In *EC*, pp. 125–131. ACM, 2004.

Max, S., MohammadTaghi, H., Debmalya, P., and Mohammad, K. Online algorithms for the santa claus problem. In *NeurIPS*, pp. 30732–30743, 2022.

Steihaus, H. The problem of fair division. *Econometrica*, 16:101–104, 1948.

Tan, Z. and Wu, Y. Optimal semi-online algorithms for machine covering. *Theor. Comput. Sci.*, 372(1):69–80, 2007.

Walsh, T. Allocation in practice. In *KI*, volume 8736 of *Lecture Notes in Computer Science*, pp. 13–24. Springer, 2014.

Walsh, T. Challenges in resource and cost allocation. In *AAAI*, pp. 4073–4077. AAAI Press, 2015.

Wu, Y., Tan, Z., and Yang, Q. Optimal semi-online scheduling algorithms on a small number of machines. In *ESCAPE*, volume 4614 of *Lecture Notes in Computer Science*, pp. 504–515. Springer, 2007.

Zeng, D. and Psomas, A. Fairness-efficiency tradeoffs in dynamic fair division. In *EC*, pp. 911–912. ACM, 2020.