A Justifications on the Normalization Assumption

In this section, we justify the normalization assumption of our model. We show that without the normalization assumption, even for the case of $n = 2$,

- no algorithm can do strictly better than 0-competitive for the allocation of goods;
- no algorithm can do strictly better than 2-competitive for the allocation of chores.

We first show the hard instance for the allocation of goods. Recall from Theorem 3.1 that with the normalization assumption, no algorithm can do strictly better than 0-competitive when $n \geq 3$. Thus it suffices to consider the case when $n = 2$.

Theorem A.1. *For the allocation of goods, without the normalization assumption, no online algorithm has competitive ratio strictly larger than* 0 *when* $n = 2$.

Proof. Assume otherwise and there exists a γ-competitive algorithm for approximating MMS allocation for $n = 2$ agents, where $\gamma \in (0, 1]$. Let r be a sufficiently large integer such that $\gamma > 1/r$. We show that in the allocation returned by the algorithm for the following instance, at least one agent $i \in N$ is allocated a bundle X_i with $v_i(X_i) < \gamma \cdot \mathsf{MMS}_i$, which is a contradiction.

Let the first item be e_1 with $v_1(e_1) = v_2(e_1) = 1$ and assume w.l.o.g. that agent 1 receives it. Then let the second item e_2 with $v_1(e_2) = r, v_2(e_2) = 1/r$. Since the algorithm is γ-competitive, e_2 should be assigned to agent 2 as otherwise for the instance with only two items $\{e_1, e_2\}$, we have $X_2 = \emptyset$, which leads to a 0-MMS allocation. Let X'_i be the bundle agent i holds before item e_3 comes.

	e_1	e_2	e_3
1	1	r	r
2	1	1/r	1

Table 9: Hard instance for goods without the normalization assumption.

For the instance shown in Table 9, the last item e_3 holds $v_1(e_3) = r, v_2(e_3) = 1$. We have

$$\mathsf{MMS}_1 = r, \mathsf{MMS}_2 = 1, \quad \text{and} \quad v_1(X'_1) = 1, v_2(X'_2) = 1/r,$$

where $v_1(X'_1) = (1/r)\cdot\mathsf{MMS}_1 < \gamma\cdot\mathsf{MMS}_1, v_2(X'_2) = (1/r)\cdot\mathsf{MMS}_2 < \gamma\cdot\mathsf{MMS}_2$. Since there must exist one agent in $\{1, 2\}$ that does not receive item e_3, the final allocation is not γ-MMS, which is a contradiction. □

Next, we show that without the normalization assumption, no online algorithm can guarantee a competitive ratio strictly smaller than 2 when $n = 2$. Note that any allocation is trivially 2-MMS when $n = 2$.

Theorem A.2. *For the allocation of chores, without the normalization assumption, no online algorithm has competitive ratio strictly smaller than* 2 *when* $n = 2$.

Proof. Assume otherwise and there exists an online algorithm with a competitive ratio $2 - \gamma$ for some constant $\gamma > 0$. Let ϵ be sufficiently small such that $\gamma > 2\epsilon$. We first construct the following instance (see Table 10), and show that the algorithm can not allocate all items $e_1, \cdots, e_l$ to agent 1, for $l = \lceil 1/\epsilon \rceil$.

Specifically, the first item e_1 has $c_1(e_1) = c_2(e_1) = 1$, and we can assume w.l.o.g. that agent 1 receives it. For all $2 \leq i \leq l$, we have

$$c_1(e_i) = \epsilon, \quad c_2(e_i) = \epsilon^{-(i-1)}.$$

	e_1	e_2	e_3	e_4	$\cdots$	e_l
1	$\boxed{1}$	ϵ	ϵ	ϵ	$\cdots$	ϵ
2	1	ϵ^{-1}	ϵ^{-2}	ϵ^{-3}	$\cdots$	$\epsilon^{-(l-1)}$

Table 10: Hard instance for chores without the normalization assumption ($n = 2$).

For the sake of contradiction, we assume all items $\{e_1, \cdots, e_l\}$ are allocated to agent 1. Then for the instance with exactly l items, we have

$$c_1(X_1) = 1 + (\lceil 1/\epsilon \rceil - 1) \cdot \epsilon \geq 2 - \epsilon, \quad \mathsf{MMS}_1 = 1,$$

which contradicts the assumption that the algorithm has a competitive ratio $2-\gamma$. Hence the algorithm must allocate some items in $\{e_2, \cdots, e_l\}$ to agent 2. Let e_p be the first item allocated to agent 2, where $1 < p \leq l$. Then we consider another instance whose first p items are the same as in Table 10, but has different future items. Let the next (and last) item be e_{p+1} (see Table 11).

	e_1	e_2	$\cdots$	e_{p-1}	e_p	e_{p+1}
1	$\boxed{1}$	$\boxed{\epsilon}$	$\cdots$	$\boxed{\epsilon}$	ϵ	$1 + (p-1) \cdot \epsilon$
2	1	ϵ^{-1}	$\cdots$	$\epsilon^{-(p-2)}$	$\boxed{\epsilon^{-(p-1)}}$	$\epsilon^{-(p-1)}$

Table 11: Hard instance for chores without the normalization assumption ($n = 2$).

Let X_i be the bundle that agent i holds before item e_{p+1} comes. Then we have

$$\begin{aligned} c_1(X_1) &= 1 + (p-2) \cdot \epsilon, \quad c_2(X_2) = \epsilon^{-(p-1)}, \\ \mathsf{MMS}_1 &= 1 + (p-1) \cdot \epsilon, \quad \mathsf{MMS}_2 = \epsilon^{-(p-1)} + \epsilon^{-(p-2)}. \end{aligned}$$

Hence we have

$$\begin{aligned} &c_1(X_1 + e_{p+1}) > (2-\epsilon) \cdot \mathsf{MMS}_1, \\ &c_2(X_2 + e_{p+1}) = 2 \cdot \epsilon^{-(p-1)} = \frac{2}{1+\epsilon} \cdot \mathsf{MMS}_2 > (2 - 2\epsilon) \cdot \mathsf{MMS}_2. \end{aligned}$$

The last equality holds because $1 > 1 - \epsilon^2$. Hence no matter which agent in $\{1, 2\}$ receives item e_{p+1}, the allocation is not $(2-2\epsilon)$-MMS to her. □

The main idea of our construction are: 1) The first item e_1 assigned to the agent 1, costs exactly MMS_1 to her at this moment, and each following coming item costs $\epsilon \cdot \mathsf{MMS}_1$ to her; 2) For agent 2, the cost of coming item increases exponentially, i.e. $c_2(e_i) = \epsilon^{-(i-1)}$ until she receives her first item. For agent 1, although the items (other than the first one) cost extremely small to her, she cannot receive all of them because otherwise, the allocation becomes 2-MMS. Hence the algorithm has to allocate at least one item to agent 2 before the $\lceil 1/\epsilon \rceil$-th item arrives. Then consider the first item e_p that agent 2 receives. Then we construct another instance in which the next item e_{p+1} satisfies $c_i(X_i + e_{p+1}) \geq (2-2\epsilon) \cdot \mathsf{MMS}_i$ for both $i \in N$, for which the algorithm can not guarantee a competitive ratio of $2-\gamma$.

Following this idea, we can extend our result to $n \geq 3$ agents straightforwardly. Since the proof is almost identical (except that we need to construct more items), we only provide a proof sketch here.

Theorem A.3. *For the allocation of chores with $n \geq 3$ agents, without the normalization assumption, no online algorithm has competitive ratio strictly smaller than 2.*

Proof Sketch. We construct a collection of instances that maintain the following properties: 1) The first item that each agent i receives costs exactly MMS_i to her, and each of the following items costs $\epsilon \cdot \mathsf{MMS}_i$ to her; 2) For those agents who have not received any item, the cost of coming items increase exponentially. We call that the first item each agent receives a large item to her, and after that each item is small. Note that any agent can receive at most one large item and $\lceil 1/\epsilon \rceil - 2$ small items, as otherwise the items she received cost more than $(2-2\epsilon) \cdot \mathsf{MMS}_i$ to her. Hence $n-1$ agents receive at most $(n-1) \cdot (\lceil 1/\epsilon \rceil - 1)$ items, and the algorithm should allocate each agent at least one item before the $\lceil (n-1)/\epsilon \rceil$-th item arrives. Consider the time when each agent has received at least one item, i.e., $X_i \neq \emptyset$ for all $i \in N$. Then we can construct the last item (similar to e_{p+1} in Table 11) such that no matter which agent receives it, the allocation is not $(2-2\epsilon)$-MMS to her, e.g., by making $c_i(e_{p+1}) \approx c_i(X_i)$ for all $i \in N$. □

B Missing Proofs of Theorem 3.1

In this section we complete the proof of Theorem 3.1 when there are $n \geq 4$ agents. Similar to our result for $n = 3$, we construct a collection of instances showing that no online algorithm can guarantee a competitive ratio strictly larger than 0 on these instances.

Lemma B.1. *No online algorithm has competitive ratio strictly larger than 0 for approximating MMS allocations for goods, for any $n \geq 4$.*

Proof. For the sake of contradiction, suppose there exists a γ-competitive algorithm for approximating MMS allocation for $n \geq 4$ 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 $v_i(X_i) < (1/r) \cdot \mathsf{MMS}_i$, which is a contradiction. Recall that for deterministic allocation algorithms, we can construct an instance adaptively depending on how the previous items are allocated.

Before we construct the instances, we first show an observation that if at some time t there are k agents with $v_i(X'_i) < (1/r) \cdot \mathsf{MMS}_i$ (where X'_i is the bundle agent i holds at time t) while there are less than k items to be arrived, then the allocation returned by the algorithm will not be γ-MMS.

Observation B.2. *If at some time t during the allocation, there are k agents K such that for all $i \in K$ we have $v_i(X'_i) < (1/r) \cdot \mathsf{MMS}_i$, and there are at most $k-1$ items to be arrived, then the final allocation returned by the algorithm is not γ-MMS.*

Proof. Since there are at most $k-1$ items that arrive after time t, some agent $i \in K$ will receive no item after time t, i.e. $X_i = X'_i$. Hence in the final allocation we have $v_i(X_i) < 1/r \cdot \mathsf{MMS}_i$ for some $i \in K$ and the allocation returned by the algorithm is not γ-MMS. □

Based on this observation we argue that any algorithm with a competitive ratio greater than 0 should assign the first $n-1$ items to $n-1$ different agents (unless some agent has value 0 on some item).

Claim B.1. *Assume that for all agent $i \in N$ and item $e \in M$ we have $v_i(e) > 0$, and $m \geq n$. Then any algorithm with competitive ratio greater than 0 should assign the first $n-1$ items $\{e_1, \cdots, e_{n-1}\}$ to $n-1$ different agents.*

Proof. Assume otherwise and some items e_i, e_j $(i \neq j)$ are assigned to the same agent. We can assume w.l.o.g. that agent $n-1$ and n receive nothing at the time when item e_n arrives, i.e. $X'_{n-1} = X'_n = \emptyset$. We show that the algorithm fails in the instances with n items. Let the last item be e_n such that $v_i(e_n) = v_i(M) - \sum_{j=1}^{n-1} v_i(e_j) > 0$ for all $i \in N$. Then for both agents $n-1$ and n we have $v_{n-1}(X'_{n-1}) = v_n(X'_n) = 0$, while $\mathsf{MMS}_{n-1} > 0, \mathsf{MMS}_n > 0$. Since there is only one item e_n to be allocated, from Observation B.2, the allocation returned by the algorithm is not γ-MMS, which is a contradiction. □

Next, we construct the instances for which no online algorithm can be γ-competitive. For each item e_j, $1 \le j \le n-1$, we let (see Table 12) $v_i(e_j) = r^3\epsilon$ for all $i < j$ and $v_i(e_j) = \epsilon$ for all $i \ge j$.

Recall that the first $n-1$ items must be assigned to $n-1$ different agents. Since when item i arrives, the valuation functions of agents $\{i, i+1, \ldots, n\}$ are identical, we assume w.l.o.g. that agent i receives item e_i for all $i \in \{1, 2, \cdots, n-1\}$. Then we consider the next item e_n with

$$v_i(e_n) = \begin{cases} r\epsilon, & \text{if } i = 1 \\ r^3\epsilon, & \text{if } 2 \le i \le n-1 \\ \epsilon, & \text{if } i = n. \end{cases}$$

We first show that any γ-competitive algorithm must assign e_n to an agent in $\{1, n\}$. Assume otherwise, i.e., $e_n \in X_i$ for some $i \notin \{1, n\}$. Then we consider the instance with $n+1$ items (as shown in Table 12), where the last item e_{n+1} has

$$v_i(e_{n+1}) = \begin{cases} 3 - (n-2)r^3\epsilon - r\epsilon - \epsilon, & \text{if } i = 1 \\ 3 - (n-i)r^3\epsilon - i\epsilon, & \text{if } 2 \le i \le n \\ 3 - (n+1)\epsilon, & \text{if } i = n. \end{cases}$$

	e_1	e_2	e_3	$\cdots$	e_{n-1}	e_n	e_{n+1}
1	$\boxed{\epsilon}$	$r^3\epsilon$	$r^3\epsilon$	$\cdots$	$r^3\epsilon$	$r\epsilon$	$3-(n-2)r^3\epsilon - r\epsilon - \epsilon$
2	ϵ	$\boxed{\epsilon}$	$r^3\epsilon$	$\cdots$	$r^3\epsilon$	$r^3\epsilon$	$3-(n-2)r^3\epsilon - 2\epsilon$
3	ϵ	ϵ	$\boxed{\epsilon}$	$\cdots$	$r^3\epsilon$	$r^3\epsilon$	$3-(n-3)r^3\epsilon - 3\epsilon$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\mathbf{n-1}$	ϵ	ϵ	ϵ	$\cdots$	$\boxed{\epsilon}$	$r^3\epsilon$	$3-r^3\epsilon-(n-1)\epsilon$
n	ϵ	ϵ	ϵ	$\cdots$	ϵ	ϵ	$3-n\epsilon$

Table 12: Instance showing why item e_n must be assigned to one of the agents in $\{1, 2\}$.

Observe that we have $X_1' = \{e_1\}, X_n' = \emptyset$ and $\mathsf{MMS}_1 = \epsilon + r\epsilon, \mathsf{MMS}_n > 0$ while there is only one item e_{n+1} that is not allocated. From Observation B.2, we have a contradiction. Hence every γ-competitive algorithm must allocate item e_n to either agent 1 or n. Depending on which agent receives item e_n, we construct two different instances. We argue that for both instances, the allocation returned by the algorithm is not γ-MMS.

For the case when agent 1 receives item e_n, we construct the instance with $n+2$ items as in Table 13.

	e_1	e_2	e_3	$\cdots$	e_{n-1}	e_n	e_{n+1}	e_{n+2}
1	$\boxed{\epsilon}$	$r^3\epsilon$	$r^3\epsilon$	$\cdots$	$r^3\epsilon$	$\boxed{r\epsilon}$	$r^3\epsilon$	$3-(n-1)r^3\epsilon - r\epsilon - \epsilon$
2	ϵ	$\boxed{\epsilon}$	$r^3\epsilon$	$\cdots$	$r^3\epsilon$	$r^3\epsilon$	$r^3\epsilon$	$3-(n-1)r^3\epsilon - 2\epsilon$
3	ϵ	ϵ	$\boxed{\epsilon}$	$\cdots$	$r^3\epsilon$	$r^3\epsilon$	$r^3\epsilon$	$3-(n-2)r^3\epsilon - 3\epsilon$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\mathbf{n-1}$	ϵ	ϵ	ϵ	$\cdots$	$\boxed{\epsilon}$	$r^3\epsilon$	$r^3\epsilon$	$3-2r^3\epsilon-(n-1)\epsilon$
n	ϵ	ϵ	ϵ	$\cdots$	ϵ	ϵ	ϵ	$3-(n+1)\epsilon$

Table 13: Instance with $n+2$ items when item e_n is assigned to agent 1.

Note that for any $i \in \{1, 2, n\}$, we have $v_i(X_i') < (1/r) \cdot \mathsf{MMS}_i$, where X_i' is the bundle agent i holds before e_{n+1} arrives, because

$$\mathsf{MMS}_1 = \mathsf{MMS}_2 = r^3\epsilon, \quad \mathsf{MMS}_n = \epsilon,$$
$$v_1(X_1') = r\epsilon + \epsilon, \quad v_2(X_2') = \epsilon, \quad v_n(X_n') = 0.$$

Since there are only two items $\{e_{n+1}, e_{n+2}\}$ to be allocated, from Observation B.2, the allocation returned by the algorithm is not γ-MMS.

Next we consider the case when agent n receives item e_n and let the next item be e_{n+1} with

$$v_i(e_{n+1}) = \begin{cases} \epsilon, & \text{if } i \in \{1, n\} \\ r\epsilon, & \text{if } i = 2 \\ r^3\epsilon, & \text{if } 3 \le i \le n-1. \end{cases}$$

We argue that item e_{n+1} must be assigned to agent 1 or 2. Assume otherwise, i.e., item e_{n+1} is assigned to some agent $j \notin \{1, 2\}$. For any $i \in N$, let X'_i be the bundle agent i holds before item e_{n+2} comes. For the instance with $n+2$ items shown in Table 14, we have

$$\mathsf{MMS}_1 = \mathsf{MMS}_2 = r\epsilon + 2\epsilon, \quad v_1(X'_1) = v_2(X'_2) = \epsilon,$$

where $v_1(X'_1) < (1/r) \cdot \mathsf{MMS}_1$ and $v_2(X'_2) < (1/r) \cdot \mathsf{MMS}_2$. Since there must exist at least one agent in $\{1, 2\}$ that does not receive item e_{n+2}, we have $X_1 = X'_1$ or $X_2 = X'_2$ in the final allocation. Therefore the allocation is not γ-MMS.

	e_1	e_2	e_3	$\cdots$	e_{n-1}	e_n	e_{n+1}	e_{n+2}
$\mathbf{1}$	$\boxed{\epsilon}$	$r^3\epsilon$	$r^3\epsilon$	$\cdots$	$r^3\epsilon$	$r\epsilon$	ϵ	$3-(n-2)r^3\epsilon - r\epsilon - 2\epsilon$
$\mathbf{2}$	ϵ	$\boxed{\epsilon}$	$r^3\epsilon$	$\cdots$	$r^3\epsilon$	$r^3\epsilon$	$r\epsilon$	$3-(n-2)r^3\epsilon - r\epsilon - 2\epsilon$
$\mathbf{3}$	ϵ	ϵ	$\boxed{\epsilon}$	$\cdots$	$r^3\epsilon$	$r^3\epsilon$	$r^3\epsilon$	$3-(n-2)r^3\epsilon - 3\epsilon$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\mathbf{n-1}$	ϵ	ϵ	ϵ	$\cdots$	$\boxed{\epsilon}$	$r^3\epsilon$	$r^3\epsilon$	$3-2r^3\epsilon-(n-1)\epsilon$
$\mathbf{n}$	ϵ	ϵ	ϵ	$\cdots$	ϵ	$\boxed{\epsilon}$	ϵ	$3-(n+1)\epsilon$

Table 14: Instance showing why item e_{n+1} must be assigned to one of the agents in $\{1, 2\}$.

Hence the algorithm must allocate item e_{n+1} to agent 1 or 2, for which case we construct the instance with $n+3$ items shown in Table 15. Note that for this instance we have

$$\mathsf{MMS}_1 = \mathsf{MMS}_2 = \mathsf{MMS}_3 = r^3\epsilon.$$

	e_1	e_2	e_3	$\cdots$	e_{n-1}	e_n	e_{n+1}	e_{n+2}	e_{n+3}
$\mathbf{1}$	$\boxed{\epsilon}$	$r^3\epsilon$	$r^3\epsilon$	$\cdots$	$r^3\epsilon$	$r\epsilon$	ϵ	$r^3\epsilon$	$3-(n-1)r^3\epsilon - r\epsilon - 2\epsilon$
$\mathbf{2}$	ϵ	$\boxed{\epsilon}$	$r^3\epsilon$	$\cdots$	$r^3\epsilon$	$r^3\epsilon$	$r\epsilon$	$r^3\epsilon$	$3-(n-1)r^3\epsilon - r\epsilon - 2\epsilon$
$\mathbf{3}$	ϵ	ϵ	$\boxed{\epsilon}$	$\cdots$	$r^3\epsilon$	$r^3\epsilon$	$r^3\epsilon$	$r^3\epsilon$	$3-(n-1)r^3\epsilon - 3\epsilon$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\mathbf{n-1}$	ϵ	ϵ	ϵ	$\cdots$	$\boxed{\epsilon}$	$r^3\epsilon$	$r^3\epsilon$	$r^3\epsilon$	$3-3r^3\epsilon-(n-1)\epsilon$
$\mathbf{n}$	ϵ	ϵ	ϵ	$\cdots$	ϵ	$\boxed{\epsilon}$	ϵ	ϵ	$3-(n+2)\epsilon$

Table 15: Assume that item e_{n+1} is allocated to one of the agents in $\{1, 2\}$.

On the other hand, (no matter which agent receives item e_{n+1}) we have

$$v_1(X'_1) \le 2\epsilon, \quad v_2(X'_2) \le r\epsilon + \epsilon, \quad v_3(X'_3) = \epsilon.$$

Since only two items e_{n+2}, e_{n+3} are not allocated, by Observation B.2, the allocation is not γ-MMS. □

C Experimental Evaluation

In this section, we perform the experimental evaluation of our algorithms on several generated datasets. For a more meaningful demonstration, we only consider the case with $n \geq 3$ agents. Therefore, we consider our 0.5-competitive algorithm for monotone instances for the allocation of goods and the $(2-1/n)$-competitive algorithm for the allocation of chores. We compare the approximation ratios of the allocations returned by our algorithms and the greedy algorithm, which greedily allocates each item e to the agent with maximum value (resp. minimum cost) on e. We generate random datasets with different sizes. The small datasets contain 10 agents and 100 items; the large datasets contain 50 agents and 500 items. For each size, we randomly generate 1000 instances. Recall that for any $i \in N$, we normalize $v_i(M) = n$ for goods and $c_i(M) = n$ for chores. To ensure that $\mathsf{MMS}_i = 1$ for any $i \in N$, we adopt the following generating method. Imagine that each agent has one item of value/cost n. We break this item into m items by first dividing it into n items, each with value/cost 1, and then randomly break each of these n items into m/n items. We sort the generated m items in descending order of values/costs. Hence, for all agent $i \in N$, we have $v_i(e_1) \geq v_i(e_2) \geq \cdots \geq v_i(e_m)$ and $\mathsf{MMS}_i = 1$ for goods, $c_i(e_1) \geq c_i(e_2) \geq \cdots \geq c_i(e_m)$ and $\mathsf{MMS}_i = 1$ for chores. In experiments for goods, the items arrive in the order of $\{e_1, \ldots, e_m\}$, i.e., as in the monotone instances. For the allocation of chores, the arrival order is random.

For the allocation of goods, we compare the performances of the greedy algorithm and our algorithm in Table 16. We report the average and minimum approximation ratios returned by the greedy algorithm and our algorithm over the 1000 instances as well as the total running time for both algorithms. It can be seen that the performance of the greedy algorithm is significantly worse than our algorithm in both small and large datasets. The average approximation ratios returned by the greedy algorithm in small and large datasets are 0.113 and 0, respectively, while those returned by our algorithm are 0.532 and 0.514, respectively. Indeed, the greedy algorithm returns a 0-MMS allocation for more than 40% instances, for small dataset. In large dataset, the greedy algorithm returns a 0-MMS allocation for more than 99% instances. In contrast, the approximation ratios of the allocations returned by our algorithm under both datasets are at least 0.5, which matches our theoretical guarantee for monotone instances (Theorem 3.5). Furthermore, our algorithm has a slight advantage in running time due to inactivation operation, which reduces the inactive agents for receiving items. This set of experiments demonstrate the effectiveness of our algorithm, which computes an approximately MMS allocation with an approximation ratio at least that from the theoretical guarantee (0.5) in each dataset.

Algorithm	Small			Large		
	Avg.	Min.	Time (s)	Avg.	Min.	Time (s)
Greedy	0.113	0	0.13	0	0	2.05
Ours	0.532	0.5	0.09	0.514	0.5	0.93

Table 16: Approximation ratios of the allocations returned by different algorithms for goods in different cases. (Min. means the minimum ratio and Avg. means the average ratio, over the 1000 instances). We also report the total running time (in seconds) for computing the allocations by each algorithm.

For the allocation of chores, the results are shown in Table 17. We report the average and maximum approximation ratios returned by the greedy algorithm and our algorithm over the 1000 instances as well as the total running time for both algorithms. Recall from our theoretical analysis that the competitive ratio of our algorithm is at most 1.9 for small dataset and 1.99 for large dataset. From Table 17, we observe that the approximation ratios of the allocations returned by our algorithm under both datasets are around 1.2, and the maximum approximation ratios (1.28 for the small dataset and 1.39 for the large dataset) are evidently better than the theoretical guarantees. In comparison, the average approximation ratio of the greedy algorithm is

2.33 in the small dataset and grows to 7.96 in the large dataset. The difference between our algorithm and the greedy algorithm is that our algorithm only allocates items to the active agents, i.e., those who receive items with cost less than the threshold ($c_i(X_i) \leq 1 - 1/n$). The experiment shows that this is crucial in achieving good approximation ratios in practice.

Algorithm	Small			Large		
	Avg.	Max.	Time (s)	Avg.	Max.	Time (s)
Greedy	2.33	4.42	0.13	7.96	17.99	0.26
Ours	1.09	1.28	2.2	1.26	1.39	4.8

Table 17: Approximation ratios of the allocations returned by different algorithms for chores in different cases. (Max. means the maximum ratio and Avg. means the average ratio in 1000 examples)

To present the performance difference between the greedy algorithm and our algorithm in more detail, we choose the small dataset for chores (for which the greedy algorithm performs the best). We plot the cumulative distribution function of approximation ratios returned by the two algorithms in Figure 1. It can be seen that our algorithm guarantees an approximation ratio smaller than 2 for all instances, while only around 20% instances returned by the greedy algorithm have an approximation ratio no more than 2. For about half of the instances, the greedy algorithm computes an allocation with an approximation ratio exceeding 2.5 with respect to MMS.

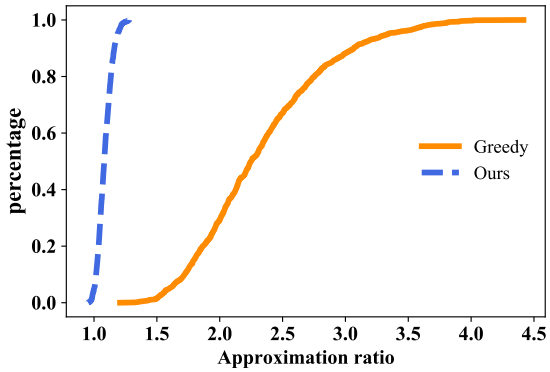

Figure 1: Cumulative distribution function of the approximation ratios of allocations returned by the two algorithms.