Approximating MMS and (symmetric) APS under Cardinality Constraints: Goods and Bads

Arjun Aggarwal, Kyra Gunluk and Ruta Mehta

July 2023

Abstract

1 Introduction

Discrete fair division is a fundamental problem within the social choice theory, where a set of m indivisible items needs to be allocated *fairly* among n agents where preferences of agent i is defined by a valuation function $V_i : 2^{\rightarrow}\mathbb{R}$. The items may be goods (positively valued) or chores (negatively valued). Among many fairness notions, *Maximin Share (MMS)* and *Any Price Share (APS)* are two of the most popular notions. Both of these have been studied extensively under additive valuations and for goods. In this paper, we study MMS and symmetric-APS for allocating both goods and chores to agents with cardinality constraints.

Both MMS and APS are *share based* fairness notions, where each agent is entitled to a bundle worth their *fair share*. Under MMS, this *fair share* of an agent is defined as the maximum value she can guarantee herself under the classical *cut-and-choose* mechanism when she is the cutter; she partitions the item set into n bundles and gets to pick last. Therefore, clearly, she will partition such that the value of the minimum valued bundle is maximized. If $\Pi()$ denotes the set of all allocations of among the n agents, and $(A_1 \ldots, A_n)$ denotes any allocation into n bundles, then the MMS value of agent i is defined as,

$$\text{MMS}_i = \max_{(A_1,\ldots,A_n)\in\Pi()} \min_{j\in[n]} V_i(A_j)$$

An MMS allocation is one where every agent i gets a bundle worth at least MMS_i. [**feige-aps**] introduced a stronger notion called *Any Price Share (APS)* via a pricing mechanism.[1] That is, if denotes the simplex of all price vectors

[1]We note that APS was defined with respect to asymmetric agents, where agent i has budget/weight of $b_i > 0$ while $\sum_i b_i = 1$, in this paper we focus on symmetric-APS.

for the m goods where the sum of all prices is 1 and all prices are non-negative, hence $= \{(p_1 \ldots, p_m) \mid \sum_{j=1}^{m} p_j = 1,\ p_j \geq 0\ \forall j\}$, then APS is given by,

$$\text{APS}_i = \min_{(p_1,\ldots,p_m)\in} \max_{S\subseteq M: \sum_{j\in S} p_j \leq 1/n} V_i(S)$$

[**feige-aps**] showed that $\text{APS}_i \geq \text{MMS}_i$ for all agent i as far as the valuation functions are monotone.

Allocations achieving MMS and APS shares may not exist even under additive valuations [**ProcacciaW14**, **FeigeST21**]. Therefore, the focus has been on finding approximate solutions, where in an α-APS (MMS) allocation, every agent receives a bundle worth at least α times their APS (MMS) value. This problem has been studied extensively for additive valuations and when contains only goods (see [**AmanatidisABFLMVW22survey**] for a survey and pointers), that is, $V_i(S) = \sum_{j\in S} v_{ij}$ for any $S \subset$, $V_i(S)$, and $v_{ij} \geq 0$, for all (i, j).

We consider agents with cardinality constraints. That is, each agent i has an additive valuation function V_i as well as a cardinality constraint of $k_i > 0$, with the understanding that if is a set of goods, then agent i wants no more than k_i items, and if is a set of chores then agent i needs to be allocated at least k_i items. Note that such a valuation function generalizes additive while is contained in submodular and supermodular functions for goods and chores, respectively. The latter are not as well-studied. The best bound known for goods under submodular functions are 10/27 and 1/3 for MMS and APS, while nothing is known for the chores under supermodular functions.

[**empty citation**] studied MMS under cardinality constraints, but assumed *homogeneous* cardinalities, *i.e.,* $k_i = k,\ \forall i$. They gave algorithms to find 2/3-MMS and 3/2-MMS respectively for goods and chores. For more complex valuation function. where is partitioned into sets $_1, \ldots, _d$ and each set has a separate cardinality constraint namely k^l for set $_l$, then extended the goods algorithm to find 1/2-MMS allocations.

We extend the above results on two fronts. First, we allow agents to have heterogeneous cardinality constraints, and second, we extend the results to the stronger notion of APS.

1. 1/2-factor for MMS and APS for goods
2. 3/2-factor for MMS and APS for chores
3. 1/2-MMS for more complex valuation function and ordered cardinality constraints. Do we have 1/2-APS here?

On the technical side we bring several new insights that may be of independent interest:

- *Sliding-window with Breaks.*
- *bag-filling until last shout*

- *reverse bag-filling*
- An analysis to show that round-robin is powerful and universal: it is enough to handle both MMS and APS, for both goods and chores to give weaker guarantees of 1/2 and 2 factors respectively.

2 Preliminaries for Goods

An instance of the fair allocation problem under *unequal cardinality constraints* is given by $I = \langle N, M, V, K \rangle$, where $N = \{1, 2, \dots n\}$ is the set of agents, $M = \{1, 2, \dots m\}$ is the set of goods, $V = \langle v_1, v_2, \dots v_n \rangle$ is the collection of the agents' valuation functions $v_i : 2^M \to \mathbb{R}_+$, and $K = \langle k_1, k_2, \dots k_n \rangle$ is the collection of cardinality thresholds. Without loss of generality, we assume that the agents are arranged in non-decreasing order of thresholds. Hence, $k_1 \leq k_2 \cdots \leq k_n$. We assume that the valuation functions are additive i.e. $v_i(S) = \sum_{g \in S} v_i(g)$ for any subset $S \subseteq M$. An *allocation* $A = (A_1, A_2, \dots A_n) \in \Pi_n(M)$ is an n-partition of M, with A_i being the bundle received by agent i.

Let $\text{Top}_i^j(S)$ denote the set of $\max(|S|, j)$ most valuable goods for agent i in S. If the instance is ordered, then Top_i^j is the same for every agent, hence we omit the subscript. For any agent i, we define the *final valuation function* $f_i : 2^M \to \mathbb{R}_+$ as follows:

$$f_i(S) := \sum_{g \in \text{Top}_i^{k_i}(S)} v_i(g)$$

Under the final valuation function, agent i can have value for at most k_i items in a set. Thus, these final valuation functions have the cardinality constraints "built" into them. We use these functions to define the maximin share of each agent.

Definition 2.1 (Maximin Share)**.** Let $I = \langle N, M, V, K \rangle$ be an instance of the fair allocation problem for goods under unequal cardinality constraints. For an agent i, let f_i be as defined above. The *maximin share* of i is defined as

$$\text{MMS}_i := \max_{A \in \Pi_n(M)} \min_{A_j \in A} f_i(A_j)$$

Definition 2.2 (Any Price Share)**.** Let $I = \langle N, M, V, K, B \rangle$ be an instance of the fair allocation problem for goods under unequal cardinality constraints, where $B = \langle b_1, b_2, \dots b_n \rangle$ is the set of agents' entitlements with every $b_i \geq 0$, and $\sum_{i=1}^n b_i = 1$. For an agent i, let f_i be as defined above. The *any price share* of i is defined as

$$\text{APS}_i := \min_{(p_1, p_2, \dots p_m) \in P} \max_{S \subseteq M} \left\{ f_i(S) \mid \sum_{j \in S} p_j \leq b_i \right\}$$

Where $P = \{(p_1, p_2, ..., p_m) \mid p_j \geq 0 \ \forall j \in M, \ \sum_{j \in M} p_j = 1\}$ is the set of item-price vectors that sum to 1.

Definition 2.3 (Any Price Share, dual definition)**.** Let $I = \langle N, M, V, K, B\rangle$ be an instance of the fair allocation problem for goods under unequal cardinality constraints, where $B = \langle b_1, b_2, \ldots b_n\rangle$ is the set of agents' entitlements with every $b_i \geq 0$, and $\sum_{i=1}^{n} b_i = 1$. The *any price share* of i is defined as

$$\text{APS}_i := \max z$$

Where z is subject to the following constraints:
1. $\sum_{T \subseteq M} \lambda_T = 1$
2. $\lambda_T \geq 0 \ \forall T \subseteq M$
3. $\lambda_T = 0 \ \forall T \subseteq M \ s.t. \ f_i(T) < z$
4. $\sum_{T \subseteq M : j \in T} \lambda_T \leq b_i \ \forall j \in M$

Lemma 2.1 (One-Good Reduction)**.** Let $I = \langle N, M, V, K, B\rangle$ be an ordered instance of fair division under ordered heterogeneous cardinality constraints, and with equal entitlements ($b_j = \frac{1}{n} \ \forall j \in M$). If a single item is given to an agent, and this item and agent are then removed from the instance, all remaining agents' APS remain the same or are increased.

Proof. We can prove this by showing that for any agent i, the optimal $z^* = APS_i$ is still a solution to the APS'_i for the new instance, and thus the maximal z must be greater or equal to this feasible value z^*. We will construct the new feasible solution from the old optimal solution as follows:
Given the entitlements are equal, $b_i = \frac{1}{n} \ \forall i$ in the original instance, and after the removal of item j^* and agent i^*, the entitlements will be $b'_i = \frac{1}{n-1} \ \forall i$. Additionally, in the new instance, $M' = M \setminus \{j^*\}$, so every $T' \subseteq M'$ is also a subset of M, and the remaining $T \subseteq M, T \nsubseteq M'$ are the exact sets $T' \cup j^*$.
Let λ^*_T be the value of λ_T in the optimal solution for the original problem for each $T \subseteq M$. Let $\lambda'_T = \frac{\lambda^*_T}{S}$ where $S = \sum_{K \subseteq M : j^* \notin K} \lambda^*_K = \sum_{K \subseteq M'} \lambda^*_K$, and $z' = z^*$. Note that S is strictly positive (specifically greater than $\frac{n-1}{n}$ as proven later) so this fraction is valid.
The first constraint holds true, since $\sum_{T \subseteq M'} \lambda'_T = \sum_{T \subseteq M'} \frac{\lambda^*_T}{S} = \frac{\sum_{T \subseteq M'} \lambda^*_T}{S} = 1$.
The second constraint holds true because the same constraint in the original LP implies that all $\lambda^*_T \geq 0$, and so $S = \sum_{K \subseteq M : j \notin K} \lambda^*_K \geq 0$, thus it follows that $\lambda'_T = \frac{\lambda^*_T}{S} \geq 0$.
The third constraint remains true because if $v_i(T') < z$ in the new LP, then in the original LP, $v_i(T) < z$ for $T = T'$, and by the third constraint in the original LP, this implies that $\lambda^*_T = 0$. Thus, $\lambda'_T = \frac{\lambda^*_T}{S} = 0$, satisfying the third constraint of the new LP.
The fourth constraint also holds. The original LP states that $\sum_{T \subseteq M : j \in T} \lambda^*_T \leq \frac{1}{n}$ $\forall j \in M$, thus, for $j^* \in M$, $\sum_{T \subseteq M : j^* \in T} \lambda^*_T \leq \frac{1}{n}$, and by the first constraint in the original LP, $1 = \sum_{T \subseteq M} \lambda^*_T = \sum_{T \subseteq M : j^* \in T} \lambda^*_T + \sum_{T \subseteq M : j^* \notin T} \lambda^*_T$, so $S = \sum_{T \subseteq M : j^* \notin T} \lambda^*_T = 1 - \sum_{T \subseteq M : j^* \in T} \lambda^*_T \geq 1 - \frac{1}{n} = \frac{n-1}{n}$. Thus, $\sum_{T \subseteq M' : j \in T} \lambda'_T = \sum_{T \subseteq M' : j \in T} \frac{\lambda^*_T}{S} = \frac{\sum_{T \subseteq M' : j \in T} \lambda^*_T}{\S} \leq \frac{1/n}{n-1/n} = \frac{1}{n} \frac{n}{n-1} = \frac{1}{n-1}$, thus it holds that $\sum_{T : j \in T} \lambda'_T \leq \frac{1}{n-1} = b'_i \ \forall j \in M'$.

Finally we can see that $(z', \lambda'_T \; \forall T \in M')$ is a feasible solution to the new LP, thus the optimal solution cannot have a smaller z than the original z^*, as we are maximizing, so the APS cannot have decreased. $\square$

Lemma 2.2. Let $I = \langle N, M, V, K, B \rangle$ be an ordered instance of fair division under ordered heterogeneous cardinality constraints, and with equal entitlements ($b_j = \frac{1}{n} \; \forall j \in M$). For any price vector $(p_1, p_2, \ldots, p_m)$ satisfying $p_j \geq 0$ $\forall j \in M, p_j = 0 \; \forall j \in M \setminus \text{Top}_i^{nk_i}(M)$, $\sum_{j \in M} p_j = 1$, the highest valued affordable bundle consists only of items from the top nk_i items.

$$\operatorname*{argmax}_{S \subseteq M} \left\{ f_i(S) \mid \sum_{j \in S} p_j \leq b_i \right\} \subseteq \text{Top}_i^{nk_i}(M)$$

Proof. To prove this claim we will use induction on the number of agents, n. When $n = 1$, we can see that $b_i = \frac{1}{n} = 1$, and since $\sum_{j \in M} p_j = 1$, when $S = \text{Top}_i^{k_i}(M)$, $\sum_{j \in S} p_j \leq 1 = b_i$, so $f_i(S) = v_i(\text{Top}_i^{k_i}(M))$. Since S is a bundle of the k_i best items, this is certainly the optimal bundle, and indeed consists of only $\text{Top}_i^{nk_i}(M)$ items.
Assuming the claim holds for all $1 \leq n' < n$, we now prove that the claim also holds for n.
Assume towards contradiction that this is false, and so the optimal bundle of affordable items, S^*, consist of at least one item not in the top nk_i items. Note that we assume $|S^*| \leq k_i$, as we can only value the top k_i items of any bundle $S \subseteq M$, and so excluding all items past the top k_i items does not change objective value. Call the bundle of items in S^* that are in $\text{Top}_i^{nk_i}(M)$ "S_1^*" and call the remaining items of S that are not in $\text{Top}_i^{nk_i}(M)$ "S_2^*", so we see that $S^* = S_1^* \cup S_2^*$ and $S_1^* \cap S_2^* = \emptyset$. Let $(p_1, p_2, \ldots, p_m)$ be a price vector that results in such an S. Construct a price vector for an instance of size $n-1$ as follows: Let the items $M' = M \setminus B$ where B is a set of k_i items in the top nk_i items of M whose prices sum to at least $\frac{1}{n}$. Such a set B is guaranteed to exist because if all n-partitions $(X_1, X_2, \ldots, X_n)$ of $\text{Top}_i^{nk_i}(M)$ with $|X_1| = |X_2| = \cdots = |X_n| = k_i$ had pricing such that $\sum_{j \in X_i} p_j < \frac{1}{n}$, then it would be true that $\sum_{j \in \text{Top}_i^{nk_i}(M)} p_j < 1$, however since $p_j = 0 \; \forall j \in M \setminus \text{Top}_i^{nk_i}(M)$, $\sum_{j \in M} p_j = \sum_{j \in \text{Top}_i^{nk_i}(M)} p_j = 1$, which is a contradiction. The remaining prices for items in M' will be less than $\frac{n-1}{n}$ since $\sum_{j \in \text{Top}_i^{nk_i}(M)} p_j = 1$ and $\sum_{j \in B} p_j \geq \frac{1}{n}$ implies $\sum_{j \in \text{Top}_i^{nk_i}(M) \setminus B} p_j = \sum_{j \in \text{Top}_i^{nk_i}(M')} p_j \leq 1 - \frac{1}{n} = \frac{n-1}{n}$. Let the new pricing $p'_j = p_j * \frac{n}{n-1}$ for all $j \in M'$, so now $\sum_{j \in M'} p_j \leq 1$. Since S^* contained items past the top nk_i items, $|M| > nk_i$, and so $|M'| = |M| - k_i > (n-1)k_i$, and since $|S^*| \leq k_i$, and $n \geq 2$, $(n-1) \geq 1$, so $|M'| > (n-1)k_i \geq |S^*|$. Thus, it must be true that there is an item in M' that is not in S^*. If the only such items were not in the top $(n-1)k_i$ items, then it would have to be true that S^* is exactly the top $(n-1)k_i$ items of M', in which case S^* would have had to be in the top nk_i items of M which we know to be false, so there must be an item in $\text{Top}_i^{(n-1)k_i}(M')$ that is not in S^*. Choose one

such item, $q \in \text{Top}_i^{(n-1)k_i}(M') \setminus S^*$, and set $p'_q = p'_q + 1 - \sum_{j \in M'} p_j$. Now, we have $\sum_{j \in M} p'_j = 1$. For all $j \in M' \setminus \text{Top}_i^{(n-1)k_i}(M')$, it must also be true that $j \in M \setminus \text{Top}_i^{nk_i}(M)$ since M' is exactly M with k_i top items removed. Thus, it must be true that $p_j = 0$, and so $p'_j = 0 * \frac{n}{n-1} = 0$ because the only new price otherwise adjusted is q, which is not in $M' \setminus \text{Top}_i^{(n-1)k_i}(M')$. Thus we have a price assignment $(p'_1, p'_2, \ldots, p'_{m'})$ such that $p'_j \geq 0$ $\forall j \in M', p'_j = 0$ $\forall j \in M' \setminus \text{Top}_i^{(n-1)k_i}(M')$, $\sum_{j \in M'} p'_j = 1$. Any bundle of items $S \subseteq M' \setminus \{q\}$ such that $\sum_{j \in S} p'_j \leq \frac{1}{n-1}$ satisfies $\sum_{j \in S} p_j * \frac{n}{n-1} \leq \frac{1}{n-1}$ or $\sum_{j \in S} p_j * \leq \frac{1}{n}$. A bundle S that contains q may no longer affordable. Thus, a bundle that is affordable in the $n-1$ case was also affordable in the n case. Since the valuation of the bundles remains the same, and every bundle $S \subseteq M'$ is also a subset of M, the maximally valued affordable bundle in the n case (which is still contained in M') must also be the maximally valued affordable bundle in the $n-1$ case. Finally we can see that this implies the solution to the $n-1$ case contains items from outside the top $(n-1)k_i$ items, which contradicts our inductive hypothesis. Thus, our initial assumption must be false, and the claim must also hold for n. $\square$

Lemma 2.3. Let $I = \langle N, M, V, K, B \rangle$ be an ordered instance of fair division under ordered heterogeneous cardinality constraints, and with equal entitlements ($b_j = \frac{1}{n}$ $\forall j \in M$). For any agent i, the APS value of the agent is at most their proportional share.

$$\text{APS}_i \leq \frac{1}{n} v_i(\text{Top}_i^{nk_i}(M))$$

Proof. Consider the price vector $p_j = \dfrac{v_{ij}}{v_i(\text{Top}_i^{nk_i}(M))}$ for all items j in the top nk_i items, and $p_j = 0$ for all other j. This satisfies price constraints, $p_j \geq 0$, and $\sum_{j \in M} p_j = \sum_{j \in \text{Top}_i^{nk_i}(M)} \frac{v_{ij}}{v_i(\text{Top}_i^{nk_i}(M))} = \frac{\sum_{j \in \text{Top}_i^{nk_i}(M)} v_{ij}}{v_i(\text{Top}_i^{nk_i}(M))} = \frac{v_i(\text{Top}_i^{nk_i}(M))}{v_i(\text{Top}_i^{nk_i}(M))} = 1$. Any $S \subseteq M$ that satisfies $\displaystyle\sum_{j \in S} p_j \leq b_i$, or $\sum_{j \in S \cap \text{Top}_i^{nk_i}(M)} \frac{v_{ij}}{v_i(\text{Top}_i^{nk_i}(M))} \leq \frac{1}{n}$, implies that $v_i(S \cap \text{Top}_i^{nk_i}(M)) \leq \dfrac{1}{n} v_i(\text{Top}_i^{nk_i}(M))$. By
textbfLemma 2.2, it is true that using this price vector, the optimal affordable bundle contains items only from $\text{Top}_i^{nk_i}(M)$. Thus, for $S^* = \text{argmax}_{S \subseteq M} \left\{ v_i(S) \mid \sum_{j \in S} p_j \leq b_i \right\}$, $S^* \subseteq \text{Top}_i^{nk_i}(M)$ so $S^* \cap \text{Top}_i^{nk_i}(M) = S^*$ so it is true that $v_i(S^*) \leq \dfrac{1}{n} v_i(\text{Top}_i^{nk_i}(M))$, or, $\max_{S \subseteq M} \left\{ v_i(S) \mid \sum_{j \in S} p_j \leq b_i \right\} \leq \dfrac{1}{n} v_i(\text{Top}_i^{nk_i}(M))$. Since APS takes the minimum over all price vectors, the APS value of agent i must be at most the proportional share. $\square$

3 1/2 Using Round-Robin

The Round-Robin algorithm consists of agents in a pre-determined order taking turns picking an item until all items are gone. In this particular algorithm, the order of agents will be such that if agent i comes before j then $k_i \leq k_j$. First we will re-scale every agents valuations such that their value of the top nk_i items is equal to n, and consequently then MMS and APS are both at most 1. We will then perform the One-Good Reduction, where every item valued greater than $\frac{1}{2}$ by at least one agent is given to one such agent, and the agent is satisfied given at least $\frac{1}{2} - MMS$ and $\frac{1}{2} - APS$, so they are resolved. The MMS and APS of remaining agents will never decrease in this step, by **Lemma 2.1**. Once a good and an agent are removed, we will rescale the valuations as before. We will the repeat the One-Good Reduction until all items that remain are valued less than $\frac{1}{2}$ by all remaining agents, or all agents are resolved, in which case we can randomly distribute remaining items, and all agents will receive even more value. Finally, Each agent will chose the best remaining item at their turn, so the first agent will end with items $1, n+1, 2n+1, ..., k_i n+1, ...$ as sorted by best to worst. Overall, the assignments will look like the following:

agent 1:	1	n+1	...	k_1n+1	(k_1+1)n+1		...		
agent 2:	2	n+2	...	k_1n+2	...	k_2n+2	(k_2+1)n+2	...	
				.					
				.					
				.					
agent n:	n	2n	...	(k_i+1)n	(k_i+2)n-1		...		$\|M\|$

Some agents with smaller cardinalities may have more items than they can valuate since everyone gets the same number of items, but using the final valuation function f_i we will show that this is not a problem.

From the perspective of each agent i looking at the top k_i items given to the first agent, we can see that it is valued at least 1: Each agent values the top nk_i items at n, and the k_i items given to the first agent are better than that given to the second which is better than the third and so on. Formally, if we define A^i to be the assigned items to agent i, we see that $v_i(Top_i^{k_i}(A^x)) \geq v_i(Top_i^{k_i}(A^y))$ $\forall$ agents $x < y$. Thus through contradiction if $v_i(Top_i^{k_i}(A^1)) < 1$ then all $v_i(Top_i^{k_i}(A^j)) < 1$, so the total nk_i items would valuate to less than n, which is false, so it must be true that $v_i(Top_i^{k_i}(A^1)) \geq 1$.

Now, for every agent i consider the value they are given: items $i, n+i, 2n+i, ...$ and the value they have lost: all other items in the top nk_i items. Each item that comes before i, which there are less than n of, is upper bounded by $\frac{1}{2}$, each item that comes after i is upper bounded by $v_i(i)$, each item that comes after $n+i$ is upper bounded by $v_i(n+i)$, and so on. There are n-1 items strictly between $jn+i$ and $(j{+}1)n+i$, so for each item $jn+i$ given to agent i, they lost $n-1$ items of value at most $jn+i$.

Thus the total value taken by i is $v_i(i)+v_i(n+i)+v_i(2n+i)+... = \sum_{j=0}^{k_i-1} v_i(jn+i)$

And the total value lost by i is at most $\frac{1}{2}n + (n-1)v_i(i) + (n-1)v_i(n+i) + ... = \frac{n}{2} + (n-1)\sum_{j=0}^{k_i-1} v_i(jn+i)$ since there are at most n items before i that are

upper bounded by $\frac{1}{2}$ and $n-1$ items that are upper bounded by $jn+i$ for every item $i, n+1, 2n+i, ..., k_i n+i$.
Thus, the total value of these top nk_i items is: Value taken + Value lost $\leq \sum_{j=0}^{k_i-1} v_i(jn+i) + \frac{n}{2} + (n-1)\sum_{j=0}^{k_i-1} v_i(jn+i) = \frac{n}{2} + n\sum_{j=0}^{k_i-1} v_i(jn+i)$, and since the top nk_i items are valued at n, we see that $n \leq \frac{n}{2} + n\sum_{j=0}^{k_i-1} v_i(jn+i)$, or $\frac{n}{2} \leq n\sum_{j=0}^{k_i-1} v_i(jn+i)$, or $\frac{1}{2} \leq \sum_{j=0}^{k_i-1} v_i(jn+i) =$ value taken by i. Thus every agent i takes value at least $\frac{1}{2}$, and since MMS and APS are each at most 1, a bundle of value $\frac{1}{2}$ is at least $\frac{1}{2}-$ MMS and at least $\frac{1}{2}-$ APS.

4 $\frac{1}{2}$ Using Bag-Filling

The first step of the algorithm is to re-scale every agents valuations such that their value of the top nk_i items is equal to n, and consequently then MMS and APS are both at most 1. The next step is a One-Good Reduction, which we will perform in the same way as at the start of the Round-Robin algorithm, and afterwards we will have all items valued less than $\frac{1}{2}$ by all agents. Now we will sort agents such that if agent i comes before j then $k_i \leq k_j$.
In this algorithm, each agent will only have "view" of their top nk_i items, so they will never be given the opportunity to value a bag outside that range. In addition, any bag taken will have at most k_i items from the top nk_i items for every i, to ensure the range of view of each agent decreases by k_i every iteration, so the range always equals the number of remaining agents times k_i. The bag-filling algorithm begins with n many empty bags labeled B_1 through B_n. We then iterate from $j = n$ to 1 to fill each bag. In each iteration, the bag B_j will receive the k_{max} least valuable remaining goods in the bag. If an agent values this bag at least 1/2, the bag and the agent can be resolved. Otherwise, we enter a while loop that shifts the window of included items in the bag by swapping the worst item in the bag with the worst item not yet considered. This will continue until a barrier of nk_i is hit for some i, in which case $|B_j| - k_i$ items will be left behind at the barrier, while only k_i of the items will continue shifting. This while loop terminates until some agent values the bag at least 1/2, which we prove is guaranteed eventually. Then the bag will be given to such an agent, both the agent and items will be removed from the instance, and the next iteration will begin filling the next bag. At the end of the algorithm, if there are any remaining items, they can be distributed randomly, as doing so can only increase the agents' assigned values.
Let $\mathrm{g}_{\min}(S)$ denote the least valuable good in S (same for all agents since we assume the instance is ordered) for any $S \subseteq M$

Algorithm 1 (1/2)-MMS Algorithm for Unequal Cardinality Constraints

Input: An ordered instance $I = \langle N, M, V, K \rangle$ with $\text{MMS}_i = 1$ and $v_i(j) < 1/2$ $\forall i \in N\ \forall j \in M$

```
1:  $A = (\emptyset, \emptyset, \ldots, \emptyset)$
2:  $B_1 = \{\}, B_2 = \{\} \ldots B_n = \{\}$
3:  **for** $j = n$ to 1 **do**
4:      Add the $k_{max}$ least valuable items to $B_j$
5:      **while** $f_i(B'_j) < 1/2\ \forall i \in N$ **do**
6:          UPDATEBAG($I$, $B'_j$, $j$)
7:      **end while**
8:      Find smallest $i \in N$ with $f_i(B_j) \geq 1/2$
9:      $A_i = \text{Top}^{k_i}(B_j), \quad M = M \setminus \text{Top}^{k_i}(B_j), \quad N = N \setminus \{i\}, \quad K = K \setminus \{k_i\}$
10: **end for**
11: If there are any remaining items, distribute them arbitrarily.
```

```
1: **function** UPDATEBAG($I$, $B_j$, $j$)
2:     Let $i := \min\{t \in N : (\text{Top}^{jk_t}(M)) \cap B_j \neq \phi\}$
3:     **if** $|(\text{Top}^{jk_i}(M)) \cap B_j| < k_i$ **then**
4:         Let $i' := \min\{t \in N : |(\text{Top}^{jk_t}(M)) \cap B_j| = k_t\}$
5:         Swap $\text{g}_{\min}((\text{Top}^{jk_i}(M)) \setminus B_j)$ with $\text{g}_{\min}((\text{Top}^{jk_{i'}}(M)) \cap B_j)$
6:     **else**
7:         Swap $\text{g}_{\min}((\text{Top}^{jk_i}(M)) \setminus B_j)$ with $\text{g}_{\min}((\text{Top}^{jk_i}(M)) \cap B_j)$
8:     **end if**
9: **end function**
```

Proofs:

To prove that this algorithm finds a $\frac{1}{2}-$MMS and $\frac{1}{2}-$APS allocation of goods to agents in polynomial time we must prove that (i) in every iteration a bag is assigned of value at least $\frac{1}{2}$ and (ii) the algorithm runs in polynomial time.

To prove (i), we need to ensure that at each iteration j of the for-loop, the while loop in lines 5-7 eventually terminates, or more specifically, UPDATEBAG() eventually produces a bag B_j such that some agent i' values it at least 1/2. To prove this, we can show that throughout the while-loop, the invariant $v_i(\text{Top}^{jk_i}(M)) \geq j$ holds for every agent i. This will prove that when $j = 1$, $v_i(\text{Top}^{k_i}(M)) \geq 1$, so for any remaining agent i, there is a bag of size at most k_i and value at least 1, and UPDATEBAG() will find such a bag by shifting the window in range jk_i up to the top k_i items, through the swapping in line 7.

We know this claim holds for $j = n$, since we re-scaled the values such that $v_i(\text{Top}^{nk_i}(M)) = n$

Now assume that at iteration j, $v_i(\text{Top}^{jk_i}(M)) \geq j$ for all agents i. We want to show that for the following iteration, $v_i(\text{Top}^{(j-1)k_i}(M \setminus B_j)) \geq j - 1$ for all remaining agents i.

Case 1: $\boldsymbol{B_j \cap \text{Top}^{jk_i}(M) = \phi}$

In this case, if the last k_i items are less than 1, then certainly the remaining top $(j-1)k+i$ will be valued greater than $j-1$. If the items are greater than 1, then since every other of the j-1 chunks of size k_i has better value, they will also have value greater than 1, and will all add up to have value greater than j-1.

Case 2: $B_j \cap$ Top$^{jk_i}(M) \neq \phi$

By the else of the conditional on line 4 of UPDATEBAG(), there will never be a chunk of size greater than k_i taken from the top jk_i items.

If exactly k_i items are taken, then $v_i(\text{Top}^{(j-1)k_i}(M \setminus B_j)) = v_i(\text{Top}^{jk_i}(M) \setminus B_j) = v_i(\text{Top}^{jk_i}(M)) - v_i(B_j)$ by additivity of valuation. Since at iteration j the while loop terminates the first time an agent (or agents) values the bag $\geq 1/2$, then in the iteration of the while loop one previous to termination, the bag currently is valued $< 1/2$ for all agents, so $v_i(B_j) < 1/2$. During the last while loop iteration, UPDATEBAG() only adds or swaps one good, which must be valued less than 1/2, so the bags value can increase by at most 1/2. Thus, after this iteration, $v_i(B_j) < 1/2 + 1/2 = 1$. Thus, $v_i(\text{Top}^{jk_i}(M)) - v_i(B_j) \geq j+1$.

If the assignment in iteration j has size $k < k_i$ taken from the top jk_i items, then either agent i has taken the bag or another agent has taken a smaller portion of the original B_j. In this case there are two possibilities:

If B_j has less than k_i items of the top jk_i items in it then by the conditional of UPDATEBAG() in line 3, the bag must still be at the bottom k items in the range. Thus it will again be true that the last k_i items of Top$^{jk_i}(M)$ will be removed to make Top$^{(j-1)k_i}(M \setminus B_j)$, and we have shown in **Case 1** that this will result in $v_i(\text{Top}^{(j-1)k_i}(M \setminus B_j)) \geq j-1$.

In any other case, B_j has exactly k_i items of the top jk_i items in it. We know that B_j is valued at most 1 by agent i, so it must be true that the value of the k items taken and the $k_i - k$ items in the top jk_i items that were in B_j, but were not taken, must be less than 1. In this case, the $k_i - k$ items in the jk_i range cannot be worse than the least valuable $k_i - k$ items in that range. Since Top$^{(j-1)k_i}(M \setminus B_j))$ will be the same as Top$^{jk_i}(M)$ only missing the k items in B_j and the $k_i - k$ least valuable items in the jk_i range, this removed value will be less or equal to the value that agent i had for B_j, which was less than 1. Thus since less than 1 value is removed, it is true that $v_i(\text{Top}^{(j-1)k_i}(M \setminus B_j)) \geq j-1$.

To prove that (ii) holds, we can simply observe that the algorithm consists of one for loop which iterates exactly n times, and inside it a while loop that iterates at most $\|M\|$ times, since there are at most $\|M\|$ items that can be swapped or added. Thus, the algorithm terminates in $O(mn)$ time.

5 Preliminaries for Chores

We can extend the definition of APS to chores by interpreting the price vector as rewards that agents earn for completing chores. In the case of goods, the entitlement of an agent was interpreted as their budget. Here, we interpret the entitlement as the amount of rewards the agent in required to earn. Let $R = \{(r_1, r_2, \ldots r_m) \mid r_j \geq 0 \; \forall j \in M, \sum_{j \in M} r_j = 1\}$ be the set of feasible rewards vectors. The AnyPrice share for chores can be defined as follows.

Definition 5.1 (AnyPrice Share for Chores)**.** The AnyPrice Share (APS) value of an agent i with entitlement b_i, denoted as $\mathrm{APS}_i(b_i)$ is defined as following

$$\mathrm{APS}_i := \max_{(r_1,r_2,\dots r_m)\in R} \min_{S\subseteq M} \left\{ d_i(S) \mid \sum_{j\in S} r_j \geq b_i \right\}$$

Cardinality constraints modeled the "diminishing returns" the agents experienced as they received more items as they received more goods. However, in the case of chores, an additional chore is more burdensome to an agent if they already have a lot of chores to do. To model this situation, each agent's cardinality constraint lower bounds the number of chores they receive. Note that when such cardinality constraints are imposed $|M| \geq \sum_{i\in N} k_i$ to ensure at least one feasible allocation exists.

Definition 5.2 (AnyPrice Share Under Heterogeneous Cardinality Constraints)**.** The APS value of an agent i with entitlement b_i and cardinality constraint k_i is defined as

$$APS_i := \max_{(r_1,r_2,\dots r_m)\in R} \min_{S\subseteq M} \left\{ d_i(S) \mid \sum_{j\in S} r_j \geq b_i, |S| \geq k_i \right\}$$

While the notion of APS is defined for arbitrary entitlements, in the following sections, we only consider the setting in which all agents have equal entitlements (i.e. $b_i = \frac{1}{n}$) since there are several nice properties when we restrict to this case. We prove the properties pertinent to our algorithms in the remainder of this section.

Similar to the goods case, the valuations functions of the agents can be scaled so that $d_i(M) = n$ for all any agent i and we can reduce an arbitrary instance to an ordered instance. Hence, we assume without loss of generality that the chores are ordered from highest disutility to lowest, $d_i(j_1) \geq d_i(j_2)\ \forall i \in N$, $\forall j_1, j_2 \in M$ such that $j_1 < j_2$.

Lemma 5.1. For any agent i, the APS value of the agent is at least their proportional share.

$$\mathrm{APS}_i \geq \frac{1}{n} d_i(M)$$

Proof. Consider the reward vector defined as follows $r_j = \dfrac{d_{ij}}{d_i(M)}$. Clearly, this reward vector is feasible. For any $S \subseteq M$ that satisfies the reward constraint, $\displaystyle\sum_{j\in S} r_j = \sum_{j\in S} \frac{d_{ij}}{d_i(M)} \geq \frac{1}{n}$. Hence, $d_i(S) \geq \dfrac{1}{n} d_i(M)$.

Thus, $\min_{S\subseteq M} \left\{ d_i(S) \mid \sum_{j\in S} r_j \geq b_i \right\} \geq \dfrac{1}{n} d_i(M)$. Since APS takes the maximum over all reward vectors, the APS value of agent i must be at least the proportional share. □

Note that 5.1 and $d_i(M) = n$ imply that $\text{APS}_i \geq 1$ for all agents $i \in N$.

Lemma 5.2. The APS value of any agent $i \in N$ is greater than the disutility of any single chore $j \in M$

$$\text{APS}_i \geq d_i(j) \; \forall i \in N, j \in M$$

Proof. We set the reward for chore j to 1 and 0 for any other chore. To satisfy the reward constraint, the agent must have chore j in their APS bundle. Hence, $\text{APS}_i \geq d_i(j)$. □

Lemma 5.3. Let $S_l = \{ln-l+1, ln-l \ldots ln+1\}$ for any l such that $ln+1 \leq |M|$

$$\text{APS}_i \geq d_i(S_l)$$

Proof. We assign a reward of $\frac{1}{ln+1}$ to the chores $1, 2, \ldots ln+1$ and assign a reward of 0 to the rest. Any bundle that satisfies the reward constraint must contain at least $l+1$ chores. Since S_l contains the $l+1$ chores with the least disutility and non-zero reward, all bundles satisfying the reward constraint must have a higher disutility than S_l. Hence, the APS value for agent i must be greater than the disutility of S_l □

As a direct consequence of 5.2 and 5.3, we have that $\text{APS}_i \geq d_i(1) \geq d_i(2) \cdots \geq d_i(n)$ and $\text{APS}_i \geq d_i(n) + d_i(n+1) \geq 2d_i(n+1)$. Hence, $\dfrac{\text{APS}_i}{2} \geq d_i(n+1) \geq d_i(n+2) \cdots \geq d_i(m)$.

Note that all proofs presented in this section are also valid for the APS under heterogeneous cardinality constraints.

6 2-APS for Chores Under Heterogeneous Cardinality Constraints

Using the properties of APS proved in the previous section, we present a simple round-robin-style algorithm that ensures that every agent receives at most twice their APS value. We allocate the chores to agents in rounds and in each round, every agent must pick their worst chore. We assume that $|M| \geq nk_{max}$ where $k_{max} = \max_{i \in N} k_i$, which proves sufficient to ensure that cardinality constraints of all agents are satisfied.

Theorem 6.1. *Algorithm 2* returns a 2-APS allocation under heterogeneous cardinality constraints

Proof. Since we assume that $|M| \geq nk_{max}$, there are at least k_{max} rounds of allocation. Every agent receives at least k_{max} chores and their cardinality constraint is satisfied. It remains to show that each agent values their bundle at most twice their APS value.

Consider $A_1 = \{1, n+1, \ldots (\lfloor \frac{m}{n} \rfloor - 1)n + 1\}$ and $A_n = \{n, 2n, \ldots \lfloor \frac{m}{n} \rfloor n\}$. Since every agent picks their worst available chore and agent n has to pick first in

Algorithm 2 2-APS Under Heterogeneous Cardinalities

```
Input: Scaled, Ordered Instance with $b_i = \frac{1}{n}$
1: $(A_1, A_2, \ldots A_n) \leftarrow (\emptyset, \emptyset \ldots \emptyset)$
2: while $M \neq \emptyset$ do
3:     for $i = 1$ to $n$ do
4:         Let $c = \text{argmax}_{j \in M} d_i(j)$
5:         $A_i \leftarrow A_i \cup \{c\}$
6:         $M \leftarrow M \setminus \{c\}$
7:         if $M = \emptyset$ then
8:             break
9:         end if
10:    end for
11: end while
```

every round, A_n contains the best chore (i.e. the chore with the lowest disutility) from every round. Thus, A_n is the best bundle amongst $A_1, A_2 \ldots A_n$ for all agents. By similar reasoning, A_1 is the worst. Since $A_1, A_2, \ldots A_n$ partition M, for any agent i, $nd_i(A_n) \leq d_i(A_1 \cup A_2 \cdots \cup A_n) = d_i(M) = n$. Hence, $d_i(A_n) \leq 1$. Since,

$$\begin{aligned} &d_i(n+1) \leq d_i(n), \\ &d_i(2n+1) \leq d_i(2n), \\ &\vdots \\ &d_i((\lfloor \frac{m}{n} \rfloor - 1)n + 1) \leq d_i((\lfloor \frac{m}{n} \rfloor - 1)n) \end{aligned}$$

implies $d_i(A_1 \setminus \{1\}) \leq d_i(A_n)$. Therefore,

$$\begin{aligned} d_i(A_i) &\leq d_i(A_n \cup \{1\}) \\ &\leq 1 + \text{APS}_i \\ &\leq 2\,\text{APS}_i\,. \end{aligned}$$

Hence, each agent values the bundle received by them at most $2\,\text{APS}_i$.

□